R E P U B L I Q U E F R A N Ç A I S E

FR 03/00089

Rec'd PCT/PTO 12 JUL 2004
10/501494

INPI
INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE

REC'D 31 MAR 2003
WIPO PCT

BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le 20 JAN. 2003

Pour le Directeur général de l'Institut national de la propriété industrielle
Le Chef du Département des brevets

CERTIFIED COPY OF PRIORITY DOCUMENT

Martine PLANCHE

DOCUMENT DE PRIORITÉ

PRÉSENTÉ OU TRANSMIS CONFORMÉMENT À LA RÈGLE 17.1.a) OU b)

BEST AVAILABLE COP

INSTITUT NATIONAL DE LA PROPRIETE INDUSTRIELLE
SIEGE
26 bis, rue de Saint Petersbourg
75800 PARIS cedex 08
Téléphone : 33 (1) 53 04 53 04
Télécopie : 33 (1) 42 93 59 30
www.inpi.fr

1er dépôt

INPI
INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE
26 bis, rue de Saint Pétersbourg
75800 Paris Cedex 08
Téléphone : 01 53 04 53 04 Télécopie : 01 42 94 86 54

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ
Code de la propriété intellectuelle - Livre VI

cerfa N° 11354*01

REQUÊTE EN DÉLIVRANCE 1/2

Important ! Remplir impérativement la 2ème page.

Cet imprimé est à remplir lisiblement à l'encre noire — 0B 540 W /190500

Réservé à l'INPI

REMISE DATE 07 MAI 2002
75 INPI PARIS B

0205751

N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI

DATE DE DÉPÔT ATTRIBUÉE PAR L'INPI 07 MAI 2002

Vos références pour ce dossier *(facultatif)* similpertin

1 NOM ET ADRESSE DU DEMANDEUR OU DU MANDATAIRE À QUI LA CORRESPONDANCE DOIT ÊTRE ADRESSÉE

Enrico MAÏM
17, rue Biscornet
75012 PARIS

Confirmation d'un dépôt par télécopie ☐ N° attribué par l'INPI à la télécopie

2 NATURE DE LA DEMANDE	Cochez l'une des 4 cases suivantes	
Demande de brevet	☒	
Demande de certificat d'utilité	☐	
Demande divisionnaire	☐	
Demande de brevet initiale	N°	Date / /
ou demande de certificat d'utilité initiale	N°	Date / /
Transformation d'une demande de brevet européen *Demande de brevet initiale*	☐ N°	Date / /

3 TITRE DE L'INVENTION (200 caractères ou espaces maximum)

Procédés pour identifier et manipuler des contenus pertinents vis-à-vis de contenus donnés.

4 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE FRANÇAISE	Pays ou organisation Date / / N° Pays ou organisation Date / / N° Pays ou organisation Date / / N° ☐ S'il y a d'autres priorités, cochez la case et utilisez l'imprimé «Suite»
5 DEMANDEUR	☐ S'il y a d'autres demandeurs, cochez la case et utilisez l'imprimé «Suite»
Nom ou dénomination sociale	MAÏM
Prénoms	Enrico
Forme juridique	
N° SIREN	
Code APE-NAF	
Adresse — Rue	17, rue Biscornet
Adresse — Code postal et ville	75012 PARIS
Pays	France
Nationalité	Italie
N° de téléphone *(facultatif)*	06 80 30 83 00
N° de télécopie *(facultatif)*	
Adresse électronique *(facultatif)*	enrico@glassbox.com

INPI
INSTITUT NATIONAL DE LA PROPRIÉTÉ INDUSTRIELLE

BREVET D'INVENTION
CERTIFICAT D'UTILITÉ

REQUÊTE EN DÉLIVRANCE 2/2

Réservé à l'INPI

REMISE DATE 7 MAI 2002
LIEU 75 INPI PARIS 8
N° D'ENREGISTREMENT NATIONAL ATTRIBUÉ PAR L'INPI 0205751

DB 540 W /150600

Vos références pour ce dossier : *(facultatif)*		similpertin
6 **MANDATAIRE**		
Nom		
Prénom		
Cabinet ou Société		
N °de pouvoir permanent et/ou de lien contractuel		
Adresse	Rue	
	Code postal et ville	
N° de téléphone *(facultatif)*		
N° de télécopie *(facultatif)*		
Adresse électronique *(facultatif)*		
7 **INVENTEUR (S)**		
Les inventeurs sont les demandeurs		[x] Oui [ ] Non **Dans ce cas fournir une désignation d'inventeur(s) séparée**
8 **RAPPORT DE RECHERCHE**		**Uniquement pour une demande de brevet (y compris division et transformation)**
Établissement immédiat ou établissement différé		[x] [ ]
Paiement échelonné de la redevance		**Paiement en deux versements, uniquement pour les personnes physiques** [x] Oui [ ] Non
9 **RÉDUCTION DU TAUX DES REDEVANCES**		**Uniquement pour les personnes physiques** [ ] Requise pour la première fois pour cette invention *(joindre un avis de non-imposition)* [ ] Requise antérieurement à ce dépôt *(joindre une copie de la décision d'admission pour cette invention ou indiquer sa référence)* :
Si vous avez utilisé l'imprimé «Suite», indiquez le nombre de pages jointes		

10 **SIGNATURE DU DEMANDEUR OU DU MANDATAIRE (Nom et qualité du signataire)** Enrico MAIM inventeur

VISA DE LA PRÉFECTURE OU DE L'INPI

L. GUICHET

La loi n°78-17 du 6 janvier 1978 relative à l'informatique, aux fichiers et aux libertés s'applique aux réponses faites à ce formulaire.
Elle garantit un droit d'accès et de rectification pour les données vous concernant auprès de l'INPI.

La présente invention concerne la recherche et la présentation d'informations pertinentes à l'utilisateur.

Les sources d'information les plus variées (notamment la Presse) sur le Web utilisent couramment des moyens d'indexation pour sélectionner dans leurs archives ou sur le Web des informations traitant du même sujet qu'une information donnée. L'indexation de chaque information est faite selon les mots (mots-clé) qu'il contient ou les catégories qui lui ont été associées. Cette approche permet, au mieux, de retrouver les informations ayant un contenu très proche.

La présente invention a pour but d'indexer des entités d'informations selon les adresses (liens) des page Web qui sont pertinentes par rapport à elle. Ainsi plutôt que les informations dont le contenu est très proche (voire redondant), l'invention vise à retrouver les informations nouvelles et réellement pertinentes.

De plus, elle vise à permettre d'indexer les informations non textuelles, comme les images et les animations. De ce fait elle s'appliquera particulièrement dans tous les domaines « de gôuts » où il est difficile de caractériser par des mots-clé l'intérêt qu'un internaute porte à l'information (quand par exemple elle représente une musique, un objet d'art, un plat culinaire, etc).et en particulier au domaine de la Presse.

L'invention propose un procédé pour permettre l'accès par un utilisateur à des d'entités d'informations pertinentes à partir d'une entité d'informations de départ, chaque entité d'informations étant accessible par un identifiant unique (URI), caractérisé en ce qu'il comprend les étapes suivantes :

a) prévoir au moins une entité d'informations similaire, présentant un contenu similaire à celui de l'entité de départ, et déterminer l'identifiant de la ou de chaque entité d'informations similaire, et

b) déterminer à partir du ou de chaque identifiant d'entité d'informations similaire un ensemble d'un ou plusieurs identifiants d'entités d'informations pertinentes par rapport à la ou chaque entité d'informations similaire.

Le procédé comprend en outre l'étape suivante :

c) permettre à l'utilisateur l'accès à au moins certaines informations pertinentes à partir de leurs identifiants respectifs

ainsi que l'étape suivante :

d) à partir des identifiants d'entités d'informations pertinentes et d'un ensemble donné d'entités d'informations supplémentaires, sélectionner les entités supplémentaires les plus similaires aux entités d'informations pertinentes.

De plus le procédé comprend une étape supplémentaire de tri des entités d'informations pertinentes par degré de pertinence.

L'étape de tri est précédée d'une étape de calcul d'un score de pertinence par rapport à la ou chaque entité d'informations similaires pour chacune des entités d'informations pertinentes.

De préférence :

Chaque entité d'informations est constituée par un fragment de page écrite en langage de marquage normalisé, ou par une telle page dans son ensemble et chaque identifiant est constitué par un identificateur uniforme de ressource (URI) du fragment ou de la page.

L'étape a) peut être réalisée par sélection par l'utilisateur d'une ou plusieurs entités d'informations similaires à l'entité d'informations de départ, ou par mise en œuvre d'un processus de détermination automatique d'entités d'informations similaires, ou encore par mise en œuvre d'un processus de détermination automatique d'entités d'informations similaires, suivie d'une sélection par l'utilisateur d'une ou plusieurs entités d'informations similaires parmi les entités d'informations similaires déterminées par ledit processus.

L'étape b) est réalisée par mise en œuvre d'un processus de détermination automatique d'entités d'informations pertinentes, qui comprend l'analyse d'une structure de graphe d'identifiants constituée par les identifiants d'entités d'informations et par les identifiants désignés par des liens activables par l'utilisateur contenus dans lesdites entités d'informations.

Le processus de détermination automatique d'entités d'informations pertinentes comprend les opérations suivantes :

b1) identifier un ensemble d'entités d'informations citantes constitué par toutes les entités d'informations possédant un lien désignant l'identifiant de l'entité d'informations similaire ou d'au moins l'une des entités d'informations similaires,

b2) identifier un ensemble d'entités d'informations candidates constitué par l'ensemble des entités d'informations dont les identifiants sont désignés dans d'autres liens contenus dans les entités d'informations citantes,

b3) pour chaque entité d'informations candidate, calculer un score de pertinence d'entité d'informations candidate entre ladite entité d'informations candidate et l'entité d'informations similaire ou l'ensemble d'entités d'informations similaires, sur la base de l'existence, dans les entités d'informations citantes prises individuellement, à la fois d'un lien désignant l'identifiant de l'entité d'informations candidate et d'un lien désignant l'identifiant de l'entité d'informations similaire ou les entités d'informations similaires, et sur la base également de scores de pertinence d'entité d'informations citante affectés à chacune des entités d'informations citantes,

b4) pour chaque entité d'informations citante, recalculer un score de pertinence d'entité d'informations citante sur la base de l'existence, dans l'entité d'informations citante en question, de liens vers les entités d'informations candidates et sur la base également des scores de pertinence d'entité d'informations candidate attribuées aux entités d'informations candidates à l'étape b3),

b5) répéter le cas échéant l'étape b3) et le cas échéant une ou plusieurs fois l'étape b4) puis l'étape b3), pour toutes les entités d'informations candidates et citantes, et

b6) déterminer lesdites entités d'informations pertinentes comme étant les entités d'informations candidates qui présentent les meilleurs scores de pertinence d'entité d'informations candidate.

Le processus de détermination automatique d'entités d'informations pertinentes peut aussi comprendre les opérations suivantes :

b1) identifier un ensemble d'entités d'informations citées constitué par toutes les entités d'informations identifiées dans des liens activables contenus dans l'entité d'informations similaire ou au moins l'une des entités d'informations similaires,

b2) former un ensemble d'entités d'informations candidates constitué par l'ensemble des autres entités d'informations ayant des liens activables identifiant lesdites entités citées,

b3) pour chaque entité d'informations candidate, calculer un score de pertinence d'entité d'informations candidate entre ladite entité d'informations candidate et l'entité d'informations similaire, ou l'ensemble d'entités d'informations similaires, sur la base de l'existence de liens activables contenus dans l'entité d'informations candidate et dans le ou les entités d'informations similaire(s) et identifiant les entités d'informations citées, et sur la base également de scores de pertinence d'entité d'informations citée affectés à chacune des entités d'informations citées,

b4) pour chaque entité d'informations citée, recalculer un score de pertinence d'entité d'informations citée sur la base de l'existence, dans l'entité d'informations citée en question, de liens activables identifiant cette entité d'informations citée et contenus dans les entités d'informations candidates et sur la base également des scores de pertinence d'entité d'informations candidate attribuées aux entités d'informations candidates à l'étape b3),

b5) répéter le cas échéant l'étape b3) et le cas échéant une ou plusieurs fois l'étape b4) puis l'étape b3), pour toutes les entités d'informations candidates et citées, et

b6) déterminer lesdites entités d'informations pertinentes comme étant les entités d'informations candidates qui présentent les meilleurs scores de pertinence d'entité d'informations candidate.

Le calcul de score de pertinence effectué à l'étape b3) comprend le calcul d'une pluralité de sommes de scores de pertinence d'entités d'informations citantes, chaque somme comprenant uniquement les scores de pertinences des entités d'informations citantes comprenant un lien vers une entité d'informations donnée constituée par l'entité d'informations candidate ou une entité d'informations similaire. Il comprend également le calcul d'au moins une somme de scores de pertinence d'entités d'informations citantes, chaque somme comprenant uniquement les scores de pertinence des entités d'informations citantes comprenant un lien vers l'une parmi un ensemble d'au moins deux entités d'informations données comprenant l'entité d'informations candidate et au moins une entité d'informations similaire.

L'entité d'informations de départ est accessible par l'utilisateur et l'étape b) comprend la prise en compte d'informations spécifiques à l'utilisateur.

Les informations spécifiques à l'utilisateur comprennent des identifiants d'entités d'informations mémorisés dans un historique de consultation d'entités d'informations par l'utilisateur.

Le procédé comprend en outre une étape de présentation à l'utilisateur, de manière associée, d'au moins deux éléments choisis dans le groupe comprenant :

- une présentation de l'entité d'informations de départ
- un lien activable désignant l'identifiant de l'entité de départ ;
- des présentations des entités d'informations pertinentes déterminées à l'étape c) ;
- des liens activables désignant les identifiants des entités d'informations pertinentes déterminées à l'étape c).

Il comprend en outre la présentation à l'utilisateur, de manière associée, d'au moins deux éléments choisis dans le groupe comprenant :

- une présentation de l'entité d'informations de départ ;

- un lien activable désignant l'identifiant de l'entité d'informations de départ ;

- des présentations des entités d'informations supplémentaires les plus similaires déterminées à l'étape d) ;

- des liens activables utilisant les identifiants des entités d'informations supplémentaires les plus similaires déterminées à l'étape d).

Un deuxième objet de l'invention est un procédé de composition ou de modification d'un ensemble d'informations contenant la désignation d'une entité d'informations de départ donnée, caractérisé en ce qu'il comprend les étapes suivantes :

- mettre en œuvre le procédé sur ladite entité de départ pour déterminer une ou plusieurs entités d'informations supplémentaires les plus similaires, et

- substituer à la désignation de l'entité de départ la désignation d'au moins l'une des entités supplémentaires les plus similaires.

Ce procédé est aussi caractérisé en ce que l'ensemble d'informations possède une structure comprenant un arbre avec au moins un nœud au niveau duquel la désignation d'une entité d'informations de départ est effectuée, et en ce qu'une mise en œuvre du procédé selon la revendication 3 au niveau de ce nœud comprend à l'étape b) la prise en compte d'identifiants d'entités d'informations pertinentes associés à au moins un nœud ancêtre de ce nœud dans l'arbre.

Selon un autre aspect du deuxième objet de l'invention, l'invention propose un procédé d'exécution variable d'un programme d'affichage de données comportant des opérations de lecture de données de nœuds dans une structure arborescente de données associée, caractérisé en ce que la structure de données comprend, en association avec des nœuds dont les données sont modifiables par l'utilisateur dans un mode administration, des indicateurs, et en ce que le procédé comprend les étapes suivantes :

- à chaque lecture d'une donnée dans le structure de données, détermination, par le programme d'affichage, de la présence d'un indicateur associé,

- en cas de présence d'un tel indicateur, affichage d'éléments graphiques interactifs permettant à l'utilisateur de manipuler la donnée associée.

En variante, l'invention propose un procédé d'exécution variable d'un programme d'affichage de données comportant des opérations de lecture de données de nœuds dans une structure arborescente de données associée, caractérisé en ce que programme comprend des instructions de lecture et de présentation de données correspondant à des nœuds sélectionnés, et en ce qu'il comprend les étapes suivantes :

- déterminer si un paramètre d'appel de l'exécution du programme d'affichage correspond à un mode administration,

- dans l'affirmative, à chaque lecture d'une donnée dans la structure de données, déterminer par le programme d'affichage, la présence d'un indicateur associé à cette lecture, et

- en cas de présence d'un tel indicateur, afficher des éléments graphiques interactifs permettant à l'utilisateur de manipuler la donnée associée.

Un troisième objet de l'invention est un procédé pour déterminer des scores de pertinence d'unités de texte telles que des phrases dans un document textuel, caractérisé en ce qu'il comprend les étapes suivantes :

- décomposition du document en une pluralité d'unités de texte,

- sélection d'au moins une unité de texte pertinente et d'unités de texte candidates,

- détermination de l'ensemble des mots signifiants contenus dans l'unité (ou les unités) de texte pertinente(s) et dans chacune des unités de texte candidates,

- pour chaque mot signifiant contenu dans l'unité (ou les unités) de texte pertinente(s), identification des unités de texte candidates citant ce mot signifiant, pour former un groupe d'unités de texte citantes,

- identification des unités de texte candidates contenant au moins un mot signifiant également cité dans les unités de texte citantes, pour former un groupe d'unités de texte co-citées,

- affectation aux unités de texte co-citées un score de pertinence en fonction desdites citations.

Vu sous un autre angle, il s'agit d'un procédé pour déterminer des scores de pertinence d'unités de texte telles que des phrases dans un document textuel, caractérisé en ce qu'il comprend les étapes suivantes :

- décomposition du document en une pluralité d'unités de texte,

- sélection d'au moins une unité de texte pertinente et d'unités de texte candidates,

- détermination de l'ensemble des mots signifiants contenus dans l'unité (ou les unités) de texte pertinente(s) et dans chacune des unités de texte candidates,

- pour chaque mot signifiant contenu dans l'unité (ou les unités) de texte pertinente(s), identification des unités de texte candidates comprenant ce mot signifiant, pour former un groupe d'unités de texte cités,

- identification des unités de texte candidates contenant au moins un mot signifiant également cité dans les unités de texte cités, pour former un groupe d'unités de texte co-citantes,

- affectation aux unités de texte co-citantes un score de pertinence en fonction desdites citations.

Selon un autre aspect du troisième objet de l'invention, on propose un procédé pour déterminer des scores attribués à des mots ou groupes de mots contenus dans des unités de texte telles que des phrases dans un document textuel, caractérisé en ce qu'il comprend une étape qui consiste à additionner les scores de pertinences des unités de texte co-citées dans lesquels lesdits mots se trouvent.

Table des matières

Chapitre 1 – Architecture de fragments

Introduction:

Le procédé selon le premier objet de l'invention sera décrit (dès le chapitre 4) dans le cadre d'une architecture particulière décrite dans les deux premiers chapitres. Cette architecture permet de composer des documents en important des fragments qui présentent l'avantage de rester tenus à jour. On représente les données en XML.

Dans le présent chapitre on préfère distinguer à priori et de manière permanente les éléments XML qui joueront le rôle de fragments. L'avantage de cette approche est que les applications (comme par exemple les feuilles de styles ou d'autres types de programmes) qui utilisent ces éléments XML peuvent être automatiquement adaptées pour permettre l'administration (édition et composition[1]) des fragments.

Dans le chapitre suivant on prendra au contraire une approche selon laquelle les fragments ne sont pas définis au niveau des données mais des applications.

Importer des fragment, composer des fragments qui restent tenus à jour

Rappelons qu'en XML, le contenu des sites est structuré de manière hiérarchique ; c'est une structure d'arbre.

Dans la méthode que nous proposons dans ce chapitre, certains des nœuds de l'arbre XML sont explicitement spécifiés comme étant des « fragments ». Les fragments sont les unités de données que l'on peut référencer directement et que l'on peut ainsi réutiliser (« importer ») à différents endroits d'un site ou même entre différents sites.

Les références de fragments sont mises en œuvre de manière innovante. Cette mise en œuvre permet de modifier la composition (en sous-fragments) des fragments importés, les parties non modifiées restant à jour par rapport à leurs sources.

La figure 1 illustre cet avantage (de tenue à jour) au moyen d'un exemple : dans le « Document 1 » le contenu du fragment importé du « Document 2 », et contenant les paragraphes 4, 5 et 6, reste tenu à jour, même si un de ses sous-fragments (le paragraphe 6) a été modifié ou si un nouveau sous-fragment (importé du « Document 3 ») y a été inséré. En effet, le « Nouveau Fragment », qui au départ a été inséré dans le Document 2, est automatiquement propagé dans le Document 1, au bon endroit dans le fragment importé.

Un simple attribut pour spécifier quels sont les noeuds fragments

Pour distinguer les nœuds fragments des autres nœuds XML, il suffit de spécifier un attribut fragment="yes".

Cet attribut a plusieurs rôles. En effet, il indique que

1. l'élément en question est un fragment,
2. l'élément parent est un « récipient »
3. et que tous les éléments enfants du nœud parent sont également des fragments.

[1] En règle générale, les termes administration, édition et composition sont ici interchangeables.

En outre,

4. le nœud racine d'un document contenant un fragment est aussi un fragment.

Prenons un exemple :

```
<?xml version='1.0' ?>
<PAGE>
 <HEADING>Ceci est le titre</HEADING>
 ...
 <SOMETHING>
   <BLOC fragment="yes">
     <HEADBLOC fragment="yes">Ceci est un premier sous-titre</HEADBLOC>
     <SMALLBLOC src="image1.jpg"/>
     <BIGBLOC>texte texte texte texte texte texte </BIGBLOC>
   </BLOC>
   <BLOC>
     <HEADBLOC>Ceci est un autre sous-titre</HEADBLOC>
     <SMALLBLOC src="image2.jpg"/>
     <BIGBLOC fragment="yes">autre texte autre texte autre texte </BIGBLOC>
   </BLOC>
 </SOMETHING>
 <SOMETHING>
   <SOMENODE>
      ... (fragments)
   </SOMENODE>
 </SOMETHING>
</PAGE>
```

En appliquant les règles mentionnées plus haut, on a implicitement (les ajouts sont en gras):

```
<?xml version='1.0' ?>
<PAGE fragment="yes">
 <HEADING >Ceci est le titre</HEADING>
 ...
 <SOMETHING recipient="yes">
   <BLOC      fragment="yes" recipient="yes">
     <HEADBLOC      fragment="yes">Ceci est un premier sous-titre</HEADBLOC>
     <SMALLBLOC     fragment="yes"src="image1.jpg"/>
     <BIGBLOC       fragment="yes">texte texte texte texte texte texte</BIGBLOC>
   </BLOC>
   <BLOC      fragment="yes" recipient="yes">
     <HEADBLOC      fragment="yes">Ceci est un autre sous-titre</HEADBLOC>
     <SMALLBLOC     fragment="yes"src="image2.jpg"/>
     <BIGBLOC       fragment="yes">autre texte autre texte autre texte</BIGBLOC>
   </BLOC>
 </SOMETHING>
 <SOMETHING>
   <SOMENODE recipient="yes">
```

```
        ... (fragments)
    </SOMENODE>
  </SOMETHING>
</PAGE>
```

Le système peut alors automatiquement générer des identifiants[2], par exemple:

```
<?xml version='1.0' ?>
<PAGE fragment-id="456">
 <HEADING>Ceci est le titre</HEADING>
 ...
 <SOMETHING recipient-id="1">
   <BLOC      fragment-id="458" recipient-id="1">
     <HEADBLOC      fragment-id="459">Ceci est un premier sous-titre</HEADBLOC>
     <SMALLBLOC     fragment-id="460"src="image1.jpg"/>
     <BIGBLOC       fragment-id="461">texte texte texte texte texte texte</BIGBLOC>
   </BLOC>
   <BLOC      fragment-id="462" recipient-id="1">
     <HEADBLOC      fragment-id="463">Ceci est un autre sous-titre</HEADBLOC>
     <SMALLBLOC     fragment-id="464"src="image2.jpg"/>
     <BIGBLOC>      fragment-id="465">autre texte autre texte autre texte</BIGBLOC>
   </BLOC>
 </SOMETHING>
 <SOMETHING>
   <SOMENODE recipient-id="2">
      ... (fragments)
   </SOMENODE>
 </SOMETHING>
</PAGE>
```

Structure et génération de l'Arbre de Contexte

Structure de fragments et récipients

L'arbre de contexte est une représentation interne des mêmes informations. Elle est conçue pour permettre de manipuler les fragments de manière efficace. Bien sûr, de multiples variantes de représentation interne sont possibles, notamment, on peut représenter la même structure de données dans le modèle relationnel.

La structure de l'arbre de contexte est récursivement composée de fragment/sous-fragments, les sous-fragments étant placés dans les récipients qui sont eux-mêmes représentés sous forme de sous-éléments des fragments.

Cette structure (en XML) est illustrée ci-dessous (pour le même exemple) :

```
<fragment id=456 state="created">
        <base-data>
            <PAGE>
```

[2] Noter que les identifiants des récipients ont une portée locale au fragment qui les contient, c'est pourquoi dans l'exemple ci-dessous ils ont plusieurs fois la valeur "1".

```
      <HEADING>Ceci est le titre</HEADING>
      ...
      <SOMETHING recipient-id="1"/>
      <SOMETHING>
        <SOMENODE recipient-id="2"/>
      </SOMETHING>
    </PAGE>
</base-data>
<recipients>
    <recipient  id="1">
      <fragment id="458" state="created">
          <base-data>
            <BLOC recipient-id="1"/>
          </base-data>
          <recipients>
            <recipient  id="1">
                <fragment id="459" state="created">
                  <base-data>
                    <HEADBLOC>Ceci est un premier sous-titre</HEADBLOC>
                  </base-data>
                </fragment>
                <fragment id="460" state="created">
                  <base-data>
                    <SMALLBLOC src="image1.jpg"/>
                  </base-data>
                </fragment>
                <fragment id="461" state="created">
                  <base-data>
                    <BIGBLOC>texte texte texte texte texte texte</BIGBLOC>
                  </base-data>
                </fragment>
            </recipient>
          </recipients>
      </fragment>
      <fragment id="462" state="created">
          <base-data>
            <BLOC recipient-id="1"/>
          </base-data>
          <recipients>
            <recipient  id="1">
                <fragment id="463" state="created">
                  <base-data>
                    <HEADBLOC> Ceci est un autre sous-titre</HEADBLOC>
                  </base-data>
                </fragment>
```

```
                    <fragment id="464" state="created">
                      <base-data>
                        <SMALLBLOC src="image2.jpg"/>
                      </base-data>
                    </fragment>
                    <fragment id="465" state="created">
                      <base-data>
                        <BIGBLOC>autre texte autre texte autre texte</BIGBLOC>
                      </base-data>
                    </fragment>
                  <recipient>
                </recipients>
              </fragment>
            </recipient>
            <recipient  id="2">
              <fragment id=... state="created">
                  ...
              </fragment>
                  ...
            </recipient>
          <recipients>
        </fragment>
```

On remarque notamment que, entre l'élément récipient SOMETHING et les sous-fragments qu'il contient (HEADING et BLOC), il y a une sorte d'articulation – matérialisée sous la forme d'un nœud « récipient » – qui peut être exploitée pour importer de nouveaux sous-fragments dans le même récipient. Au niveau du noeud <base-data> du fragment 456 (nœud PAGE), cette articulation est représentée par l'attribut recipient-id="1" (qui spécifie le numéro du récipient dans ce fragment), tandis qu'il est représenté par l'élément <recipient id="1"> (c'est une balise et non pas un attribut) au niveau de ses sous-fragments.

A l'étape d'affichage à l'écran (ou pour n'importe quelle autre présentation à l'utilisateur), le fragment et ses sous-fragments sont assemblés et une transformation au moyen d'une feuille de style en XSLT (ou au moyen d'un programme d'ordinateur ayant le même effet – on illustrera un tel programme plus loin) permet de présenter l'aspect graphique final de la page[3].

Importation d'un fragment : création d'un lien vers un autre fragment

Cette structure en fragments et récipients permet d'importer des fragments et de tenir chaque fragment importé à jour, même quand un (ou plusieurs) de ses sous-fragment(s) a (ont) été modifié(s) ou supprimé(s), ou encore quand un (ou plusieurs) nouveau(x) sous-fragment(s) a (ont) été importé(s) au sein du fragment importé en question.

En résultat d'une importation[4] d'un fragment source au sein d'un fragment destination, la référence du fragment source (autrement dit, le lien vers le fragment source) est insérée sous le

[3] Noter qu'un fragment peut aussi représenter directement un fragment de la page (HTML) d'origine. Dans ce cas il possède un sous-nœud « base-data » qui contient le code (HTML) du fragment d'origine.

[4] (on utilise le même mot « importation » qu'il s'agisse d'une importation de fragment au sein d'une page ou qu'il s'agisse de la création d'un lien)

nœud fragment destination en tant qu'attribut « ref » d'un sous-nœud « derived-data » du fragment destination. La mise en œuvre de cette référence peut être sous la forme d'un appel à un Service Web apte à retourner le fragment en question.[5]

Un lien est représenté en tant que fragment. En effet, ce sont les feuilles de style (par exemple en XSLT, ou autres types de programmes) associés aux (ou à certains seulement des) fragments (ou encore une feuille de style associée au document tout entier) qui déterminent la présentation des pages et déterminent en particulier s'il faut présenter ou pas le fragment qui est au bout d'un lien.[6]

On va maintenant prendre un exemple d'un deuxième document duquel on va importer un fragment dans le premier document.

Document 2 :

```
<?xml version='1.0' ?>
<PAGE fragment-id="789">
 <SOMETHING recipient-id="1">
   <BLOC      fragment-id="790" recipient-id="1">
     <HEADBLOC      fragment-id="791">Sous-titre du bloc externe</HEADBLOC>
     <SMALLBLOC     fragment-id="792"src="image3.jpg"/>
   </BLOC>
 </SOMETHING>
</PAGE>
```

Voici l'arbre de contexte qui y correspond:

Représentation interne (arbre de contexte) du Document 2 (géré par serviceweb123):

```
<fragment id=789 state="created">
        <base-data>
            <PAGE>
              <SOMETHING recipient-id="1"/>
            </PAGE>
        </base-data>
        <recipients>
            <recipient id="1">
              <fragment id="790" state="created">
                  <base-data>
                     <BLOC recipient-id="1"/>
                  </base-data>
                  <recipients>
                     <recipient id="1">
```

[5] Par ailleurs, optionnellement un sous-élément « cache » permet de stocker provisoirement le code (HTML, ou XML + éventuellement la feuille de style) du fragment source importé (pour améliorer les performances dans le cas d'importation à partir d'un serveur distant). Ceci est partiellement défini plus loin.

[6] Ainsi un fragment peut être présenté sous la forme d'un simple lien activable vers une autre page (dans laquelle le contenu du fragment est présenté), ou bien le contenu du fragment peut être présenté directement au sein de son fragment parent, ou encore un extrait en est présenté avec en plus un lien activable qui permet d'aller voir la présentation du fragment en entier.

```
                    <fragment id="791" state="created">
                      <base-data>
                        <HEADBLOC>Sous-titre du bloc externe</HEADBLOC>
                      </base-data>
                    </fragment>
                    <fragment id="792" state="created">
                      <base-data>
                        <SMALLBLOC src="image3.jpg"/>
                      </base-data>
                    </fragment>
                  </recipient>
                </recipients>
              </fragment>
            </recipient>
          <recipients>
        </fragment>
```

Ainsi, si par exemple l'utilisateur importe le fragment n° 790 du deuxième document dans le récipient n° 1 du fragment n° 456 du premier document juste avant le fragment n° 462, et qu'il supprime le fragment n° 792, on a la structure illustrée ci-dessous.

Représentation interne (arbre de contexte) du Document 1 (après l'importation):

```
<fragment id=456 state="created">
    <base-data>
      <PAGE>
       <HEADING>Ceci est le titre</HEADING>
       ...
       <SOMETHING recipient-id="1"/>
       <SOMETHING>
         <SOMENODE recipient-id="2"/>
       </SOMETHING>
      </PAGE>
    </base-data>
    <recipients>
      <recipient id="1">
        <fragment id="458" state="created">
          <base-data>
            <BLOC recipient-id="1"/>
          </base-data>
          <recipients>
            <recipient id="1">
              <fragment id="459" state="created">
                <base-data>
                  <HEADBLOC>Ceci est un premier sous-titre</HEADBLOC>
                </base-data>
```

```
                </fragment>
                <fragment id="460" state="created">
                  <base-data>
                    <SMALLBLOC src="image1.jpg"/>
                  </base-data>
                </fragment>
                <fragment id="461" state="created">
                  <base-data>
                    <BIGBLOC>texte texte texte texte texte texte</BIGBLOC>
                  </base-data>
                </fragment>
            </recipient>
        </recipients>
    </fragment>
    <fragment id="466" state="imported">
        <derived-data ref= "webservice123/GetFragment?Id=790" />
        <recipients>
          <recipient id="1">
              <fragment id="467" state="accepted">
                    <derived-data
                    ref= "webservice123/GetFragment?Id=791" />
              </fragment>
              <fragment id="468" state="suppressed">
                    <derived-data
                    ref= "webservice123/GetFragment?Id=792" />
              </fragment>
          <recipient>
        </recipients>
    </fragment>
    <fragment id="462" state="created">
        <base-data>
          <BLOC recipient-id="1"/>
        </base-data>
        <recipients>
          <recipient id="1">
              <fragment id="463" state="created">
                <base-data>
                  <HEADBLOC> Ceci est un autre sous-titre</HEADBLOC>
                </base-data>
              </fragment>
              <fragment id="464" state="created">
                <base-data>
                  <SMALLBLOC src="image2.jpg"/>
                </base-data>
              </fragment>
```

```
                    <fragment id="465" state="created">
                      <base-data>
                        <BIGBLOC>autre texte autre texte autre texte</BIGBLOC>
                      </base-data>
                    </fragment>
                  <recipient>
                </recipients>
              </fragment>
            </recipient>
            <recipient id="2">
                ...
            </recipient>
          <recipients>
        </fragment>
```

La règle qui a ici été appliquée est que les sous-fragments d'un fragment qui vient d'être importé sont « acceptés » par défaut. C'est pourquoi le sous-fragment n° 467 est à l'état accepté (noté « accepted »).

Par la suite, si dans le document 2, un sous-fragment supplémentaire est ajouté, celui-ci fera implicitement partie du fragment importé dans le document 1. On dit qu'il est à l'état suggéré (« suggested ») et quand l'utilisateur accèdera au document 2 en mode administration, il le verra présenté à l'état « suggested » et pourra le faire passer à l'état « accepted » pour le valider, auquel cas ce nouveau sous-fragment va effectivement faire partie de l'arbre de contexte du document 1.

Options

Structure de cache pour les fragments importés

Comme décrit plus haut, un fragment peut être constitué d'une référence à un autre fragment, ce dernier se trouvant éventuellement dans un autre ordinateur et pouvant lui même référencer un troisième fragment se trouvant dans un troisième ordinateur, et ainsi de suite, jusqu'au dernier fragment qui lui pointe sur une microstructure.

En conséquence, pour assembler tous les fragments récupérés au bout des chaînes de références, beaucoup de communications entre machines peuvent être nécessaires. L'assemblage du document à partir de son contexte peut donc être relativement lourd.

De plus, il suffit qu'une seule machine (ou logiciel serveur) dans une chaîne soit en panne, ou qu'une seule connexion soit rompue, pour que le fragment en bout de chaîne ne puisse pas être récupéré, c'est-à-dire téléchargé de proche en proche jusqu'au document que l'on cherche à assembler.

L'idée du cache est de mémoriser en local, dans l'arbre de contexte, les données « base-data » des fragments externes accédées, du moins certaines d'entre elles. On ne mémorisera en général que les données importées qui sont distantes (ne résident pas sur le même site).

Représentation interne (arbre de contexte) du Document 1 (après l'importation):

```
<fragment id=456 state="created">
    <base-data>
        ...
```

```
</base-data>
<recipients>
    <recipient id="1">
      <fragment id="458" state="created">
          <base-data>
            <BLOC recipient-id="1"/>
          </base-data>
          <recipients>
            <recipient id="1">
                ... (fragments n° 459 à 461)
            </recipient>
          </recipients>
      </fragment>
      <fragment id="466" state="imported">
          <derived-data ref= "webservice123/GetFragment?Id=790">
              <cached-data>
                <BLOC recipient-id="1"/>
              </cached-data>
          </derived-data>
          <recipients>
            <recipient id="1">
              <fragment id="467" state="accepted">
                  <derived-data
                  ref= "webservice123/GetFragment?Id=791">
                      <cached-data>
                    <HEADBLOC>Sous-titre du bloc externe</HEADBLOC>
                      </cached-data>
                  </derived-data>
              </fragment>
              <fragment id="468" state="suppressed">
                  <derived-data
                  ref= "webservice123/GetFragment?Id=792" />
              </fragment>
            <recipient>
          </recipients>
       </fragment>
       ... (fragments n° 462 et ses sous-fragments)
    </recipient>
    <recipient id="2">
          ...
    </recipient>
  <recipients>
</fragment>
```

Sceller un fragment

L'attribut « sealed » permet de « sceller » un fragment. Ainsi, un fragment ayant l'attribut sealed="yes" indique que quand le fragment en question est importé ailleurs, ses récipients et sous-fragments ne sont pas visibles (et ne peuvent donc pas être manipulés) ; l'ensemble du fragment est vu comme un seul bloc (c'est le cas de le dire)..

Reprenons l'exemple du deuxième document duquel on importe un fragment dans le premier document.

Document 2 :

```
<?xml version='1.0' ?>
<PAGE fragment-id="789" >
 <SOMETHING recipient-id="1">
   <BLOC      fragment-id="790" sealed="yes" recipient-id="1">
     <HEADBLOC      fragment-id="791">Sous-titre du bloc externe</HEADBLOC>
     <SMALLBLOC     fragment-id="792"src="image3.jpg"/>
   </BLOC>
 </SOMETHING>
</PAGE>
```

Voici l'arbre de contexte qui y correspond:

Représentation interne du Document 2 (géré par serviceweb123):

```
<fragment id=789 state="created">
    <base-data>
        <PAGE>
          <SOMETHING recipient-id="1"/>
        </PAGE>
    </base-data>
    <recipients>
        <recipient id="1">
          <fragment id="790" sealed="yes" state="created">
              <base-data>
                 <BLOC recipient-id="1"/>
              </base-data>
              <recipients>
                 <recipient id="1">
                     <fragment id="791" state="created">
                       <base-data>
                         <HEADBLOC>Sous-titre du bloc externe</HEADBLOC>
                       </base-data>
                     </fragment>
                     <fragment id="792" state="created">
                       <base-data>
                         <SMALLBLOC src="image3.jpg"/>
                       </base-data>
```

```
                    </fragment>
                </recipient>
            </recipients>
        </fragment>
    </recipient>
<recipients>
</fragment>
```

L'utilisateur -qui (comme avant) importe le fragment n° 790 du deuxième document dans le récipient n° 1 du fragment n° 456 du premier document juste avant le fragment n° 462- ne peut maintenant pas supprimer le fragment n° 792 car il ne le voit pas. En effet, on a maintenant la structure illustrée ci-dessous.

Représentation interne du Document 1 (après l'importation):

```
<fragment id=456 state="created">
    <base-data>
        ...
    </base-data>
    <recipients>
        <recipient id="1">
            <fragment id="458" state="created">
                ...
            </fragment>
            <fragment id="460" state="imported">
                <derived-data ref= "webservice123/GetFragment?Id=790" />
            </fragment>
            <fragment id="462" state="created">
                ...
            </fragment>
        </recipient>
        <recipient id="2">
                ...
        </recipient>
    <recipients>
</fragment>
```

Autres représentations de la structure des fragments

Représentation duale

Pour la structure interne, on peut éviter de devoir créer un arbre de contexte si on ajoute des attributs « fragment-ref » (associés aux balises).

Représentation interne duale du Document2 (webservice123) :

```
<?xml version='1.0' ?>
<PAGE fragment-id="789" state="created" >
```

```
 <SOMETHING recipient-id="1">
   <BLOC      fragment-id="790" state="created" recipient-id="1">
     <HEADBLOC
       fragment-id="791" state="created" >Sous-titre du bloc externe</HEADBLOC>
     <SMALLBLOC
       fragment-id="792" state="created" src="image3.jpg"/>
   </BLOC>
 </SOMETHING>
</PAGE>
```

Représentation interne duale du Document 1 (après l'importation):

```
<?xml version='1.0' ?>
<PAGE fragment-id="456" state="created" >
  <HEADING>Ceci est le titre</HEADING>
 ...
 <SOMETHING recipient-id="1">
   <BLOC      fragment-id="458" state="created" recipient-id="1">
     <HEADBLOC
       fragment-id="459" state="created">Ceci est un premier sous-titre</HEADBLOC>
     <SMALLBLOC
       fragment-id="460" state="created" src="image1.jpg"/>
     <BIGBLOC
       fragment-id="461" state="created">texte texte texte texte textetexte</BIGBLOC>
   </BLOC>
   <BLOC fragment-id="466" state="imported" recipient-id="1"
       fragment-ref="webservice123/GetFragment?Id=790">
     <HEADBLOC fragment-id="467" state="accepted"
       fragment-ref="webservice123/GetFragment?Id=791"/>
     < SMALLBLOC fragment-id="468" state="suppressed"
       fragment-ref ="webservice123/GetFragment?Id=792"/>
   </BLOC>
   <BLOC fragment-id="462" state="created" recipient-id="1">
     <HEADBLOC
       fragment-id="463" state="created">Ceci est un autre sous-titre</HEADBLOC>
     <SMALLBLOC
       fragment-id="464" state="created" src="image2.jpg"/>
     <BIGBLOC fragment-id="465" state="created">
       autre texte autre texte autre texte</BIGBLOC>
   </BLOC>
 </SOMETHING>
 <SOMETHING>
   <SOMENODE recipient-id="2">
       ... (fragments)
   </SOMENODE>
 </SOMETHING>
</PAGE>
```

De même on peut ajouter un attribut « cached-data » pour remplacer le sous-élément de même nom.

Représentation avant présentation à l'utilisateur

On doit générer une représentation apte à permettre l'application de la feuille de style pour présentation à l'utilisateur. Cette représentation peut avoir toutes les références déjà résolues, comme ci-dessous.

Document 1 (après l'importation) prêt à être présenté:

```
<?xml version='1.0' ?>
<PAGE fragment-id="456" state="created" >
 <HEADING>Ceci est le titre</HEADING>
 ...
 <SOMETHING recipient-id="1">
   <BLOC      fragment-id="458" state="created" recipient-id="1">
     <HEADBLOC
       fragment-id="459" state="created">Ceci est un premier sous-titre</HEADBLOC>
     <SMALLBLOC
       fragment-id="460" state="created" src="image1.jpg"/>
     <BIGBLOC
       fragment-id="461" state="created">texte texte texte texte texte texte</BIGBLOC>
   </BLOC>
   <BLOC      fragment-id="466" state="imported" recipient-id="1">
     <HEADBLOC
       fragment-id="467" state="accepted">Sous-titre du bloc externe</HEADBLOC>
   </BLOC>
   <BLOC      fragment-id="462" state="created" recipient-id="1">
     <HEADBLOC
       fragment-id="463" state="created">Ceci est un autre sous-titre</HEADBLOC>
     <SMALLBLOC
       fragment-id="464" state="created" src="image2.jpg"/>
     <BIGBLOC>
       fragment-id="465" state="created">autre texte autre texte autre texte</BIGBLOC>
   </BLOC>
 </SOMETHING>
 <SOMETHING>
   <SOMENODE recipient-id="2">
     ... (fragments)
   </SOMENODE>
 </SOMETHING>
</PAGE>
```

En variante, la feuille de style (ou le programme de présentation) peut prendre en entrée la représentation duale (qui peut être effectivement utilisée en interne ou générée à partir de l'arbre de contexte qui elle est utilisée comme représentation interne, ou encore à partir d'une base de données relationnelle par exemple).

On évite ainsi d'avoir à disposition des feuilles de style (ou programmes de présentation) qui s'appliquent sur l'arbre de contexte directement.

Composants de présentation des pages

L'arbre des composants d'une page

Le Document 1 doit par exemple être présenté sous la forme schématiquement illustrée à la figure 2.

Si les composants de présentation peuvent être créés de manière statique, leur assemblage (selon la figure) peut être effectuée simplement au moyen d'un programme ayant l'allure suivante:

```
using System;
using …;
using System.Web.UI.WebControls;

namespace Lib
{
      public class StaticCreation : WebControl
      {
            public StaticCreation() {}
            protected override void CreateChildControls()
            {
                  ctrl6 _ctrl6 = new ctrl6();

                  ctrl4 _ctrl4 = new ctrl4 ();
                  _ctrl4.Pos = 1;

                  ctrl3 _ctrl3 = new ctrl3("Ceci est un premier sous-titre");
                  _ctrl4.Add (_ctrl3);

                  ctrl1 _ctrl1 = new ctrl1();
                  _ctrl1.Image = "image1.jpg";
                  _ctrl4.Add(_ctrl1);

                  ctrl2 _ctrl2 = new ctrl2("texte texte texte texte texte texte");
                  _ctrl4.Add (_ctrl2);
                  _ctrl6.Add(_ctrl4);

                  _ctrl4 = new ctrl4 ();
                  _ctrl4.Pos = 2;

                  _ctrl3 = new ctrl3("Ceci est un autre sous-titre");
                  _ctrl4.Add (_ctrl3);

                  _ctrl1 = new ctrl1();
                  _ctrl1.Image = "image2.jpg";
                  _ctrl4.Add(_ctrl1);

                  _ctrl2 = new ctrl2("autre texte autre texte autre texte");
                  _ctrl4.Add(_ctrl2);
                  _ctrl6.Add(_ctrl4);

                  Controls.Add(_ctrl6);
            }
      }
```

```
};
```

Ici la propriété « pos » (position) est renseignée pour le composant ctrl4. Les composants sont par exemple programmés comme suit (dans le principe) :

```
public class ctrl1 : Control
{
      Image _img;
      int _algn,_pos;

      public int Pos
      {
            get { return _pos;}
            set { _pos = value; }
      }
      public String Image
      {
            get { return _img.ImageUrl; }
            set { _img.ImageUrl = value; }
      }
      public int Align
      {
            get { return _algn; }
            set { _algn = value; }
      }
      protected override void CreateChildControls()
      {
            if ( _algn == 0)
                  _img.ImageAlign = ImageAlign.Left;
            else
                  _img.ImageAlign = ImageAlign.Right;
            Controls.Add(_img);
      }
}
```

Comme le composant de présentation ctrl4 a sa position en tant que valeur d'attribut, il peut utiliser cette information pour aligner les composants enfants comme l'indique la figure : si sa position est impaire, le composant enfant ctrl1 est créé en étant aligné à gauche, sinon à droite.

```
public class recipient : Control
{
      public ArrayList _ctrls;
      public recipient() { _ctrls = new ArrayList(); }
      public void Add(Control obj)
      {
            _ctrls.Add(obj);
      }
}
public class ctrl4 : recipient
{
      int _pos;

      public ctrl4():base() { }
      public new void Add(Control obj)
      {
            base.Add(obj);
      }
      public int Pos
```

```
        {
              get { return _pos;}
              set { _pos = value; }
        }
        protected override void CreateChildControls()
        {
              Control obj;
              if (_ctrls.Count ==0 )
                    return;

              Table tbl;
              TableRow tr;
              TableCell td;

              tbl = new Table();
              tr = new TableRow();
              td = new TableCell();
              obj = (Control) _ctrls[0];
              td.Controls.Add(obj);
              tr.Cells.Add(td);
              tbl.Rows.Add(tr);
              tr = new TableRow();
              td = new TableCell();
              obj = (Control) _ctrls[1];
              ((ctrl1) obj).Align = _pos%2;
              td.Controls.Add(obj);
              obj = (Control) _ctrls[2];
              td.Controls.Add(obj);
              tr.Cells.Add(td);
              tbl.Rows.Add(tr);
              Controls.Add(tbl);
        }
  }
```

Génération dynamique de l'arbre des composants à partir des fragments

Nous décrivons maintenant le cas où les composants de présentation (dont le code source a été décrit dans la section précédente) doivent être créés dynamiquement à partir des fragments XML et de leurs composants de présentation correspondant (la correspondance étant spécifiée avec les composants FgmtCtrl).

Le procédé de présentation opère à partir des correspondances suivantes entre les fragments et les composants qui les présentent:

<PAGE **fragment-id="456"**>	(présenté par)	→	(composant)	ctrl6
<BLOC **fragment-id="458"**>		→		ctrl4
<HEADBLOC **fragment-id="459"**>		→		ctrl3
<SMALLBLOC **fragment-id="460"**>		→		ctrl1
<BIGBLOC **fragment-id="461"**>		→		ctrl2
<BLOC **fragment-id="462"**>		→		ctrl4
<HEADBLOC **fragment-id="463"**>		→		ctrl3
<SMALLBLOC **fragment-id="464"**>		→		ctrl1
<BIGBLOC **fragment-id="465"**>		→		ctrl2

Comme l'illustre la figure , ctrl4 est en charge de présenter le fragment SMALLBLOC (par le composant ctrl1) alignée à gauche quand son fragment parent (BLOC) est à une position impaire et alignée à droite sinon.

La page Web source (en l'occurrence c'est une page aspx, selon la technologie ASP.NET de Microsoft, marques déposées) sera constituée de composants FgmtCtrl chargés d'assembler les composants de présentation comme spécifié ci-dessus.

Les composants FgmtCtrl ont pour rôle de créer dynamiquement l'arbre des composants de présentation des fragments. Ils sont placés dans un composant PageCtrl auquel on spécifie quelle est la source XML. La page aspx a l'allure suivante :

```
<%@ Register TagPrefix="App" NameSpace="WebControlLibrary123" Assembly="tagcontrols" %>
<%@ Page Language="C#" Inherits="Lib.PageCtrl" %>
<html>
<body>
<form runat="server">

<App:PageCtrl Data="..\document1.xml">

  <App:FgmtCtrl Element="PAGE" Control="ctrl6" NameSpace="Lib" Asm="PWebControls" Args="Empty"
runat="server">

    <App:FgmtCtrl Element="BLOC" Control="ctrl4" NameSpace="Lib" Asm="PWebControls" Args="Empty"
runat="server">

      <App:PropertyMapping Name="Align" Value="Const:1" Asm="PWebControls" runat="server"/>

      <App:FgmtCtrl Element="HEADBLOC" Control="ctrl3" NameSpace="Lib" Asm="PWebControls" Args="Content"
runat="server"/>

      <App:FgmtCtrl Element="SMALLBLOC" Control="ctrl1" NameSpace="Lib" Asm="PWebControls" Args="Content"
runat="server">

          <App:PropertyMapping Name="Image" Value="@src" Asm="PWebControls" runat="server"/>

        </App:FgmtCtrl>

      <App:FgmtCtrl Element="BIGBLOC" Control="ctrl2" NameSpace="Lib" Asm="PWebControls" Args="Content"
runat="server" />

    </App:FgmtCtrl>

  </App:FgmtCtrl>

</App:PageCtrl>

</form>
</body>
</html>
```

Args="Empty" indique qu'aucun argument ne sera fourni au constructeur du composant en question.

Args="Content" indique que le contenu (chaîne de caractères) de l'élément sera passé en argument au constructeur.

L'application aspx commence par éxecuter le composant PageCtrl qui lui exécute le (ou les) composant(s) FgmtCtrl de premier niveau (ceux qui n'ont pas de composant FgmtCtrl parent). Chaque FgmtCtrl, pour tous les fragments dont le chemin correspond à la valeur de l'attribut « Element », crée une instance de composant de présentation spécifié par l'attribut Control, en

passant au constructeur les paramètres spécifiés par l'attribut Args, et l'ajoute à la page Web en cours de construction.

Les composants enfants PropertyMapping permettent ensuite de renseigner les valeurs des propriétés du composant créé. De manière transparente, la propriété Self du composant en question est aussi renseignée, de manière à ce que ce dernier connaisse l'adresse exacte du fragment qu'il a le rôle de présenter.

Ensuite (voir la procédure CreateTree décrite ci-après) les autres composants, générés par les autres composants FgmtCtrl, sont ajoutés au sein de leur parent. Ainsi l'arbre des composants se construit dynamiquement.

FgmtCtrl est mis en œuvre de la manière suivante (en pseudo-code) :

```
FgmtCtrl : WebControl
{
...
CreateChildControls()
    {           . . .
        for each fragment in all xml fragments that match Tag
        {
                parent = new ctrlX()
                Controls.Add(parent)
                        (CreateChildControls() is called automatically within parent)
                CreateTree(parent, this (the current FgmtCtrl ctrl, to access the children
FgmtCtrls), fragment, ...)
        }
    }

CreateTree(parent, this, fragment, ...)
{
for each FgmtCtrl ctrl (childFgmtCtrl) which is child of this
{
        // inspect its given Element and control name, ... properties
        // say Element = childtag, Control = childctrl
        for each childfragment in all xml fragments within fragment that match childtag
        {
                // create instance of the control childctrl

                child = childctrl ("Args evaluated if any")
                    (CreateChildControls() is called automatically within child)
                parent.Add(child);
                . . .
                if (children exist within childFgmtCtrl)
                        CreateTree(child, childFgmtCtrl, childfragment,...)
        }
}
```

Transformations pour permettre la maintenance

Mode « Administration »

Lorsque l'utilisateur accède à la page dans un mode « Administration » (ou « Edition », ou encore « Manipulation », etc.), par exemple en cliquant sur un bouton comme illustré à la figure 3.a, les composants de présentation prennent un comportement légèrement différent en ce sens qu'ils intercalent dans la présentation, pour chaque fragment, un ensemble de boutons et poignées/cibles (pour glisser-déposer) permettant d'effectuer des manipulations sur (ou entre) les fragments. Ceci est illustré à la figure 3.b.

Etats et Transitions des fragments

Les fragments peuvent être dans l'un des états suivants :

- créé (« created »)
- importé (« imported »)
- accepté (« accepted »)
- supprimé (« suppressed »)
- gelé (« frozen »)
- suggéré (« suggested »)

Quand un fragment est importé à partir d'un fragment source, un nouveau fragment, qui est une référence sur le fragment source, est créé à l'état 'importé'. Ensuite, les sous-fragments correspondant aux sous-fragments dudit fragment source (tous les descendants) sont également créés, sous forme de référence à ces derniers, à l'état 'accepté' par défaut et peuvent passer à l'état 'suggéré' sous l'action de l'utilisateur.

Le comportement des fragments dans les états 'importé' et 'accepté' est ensuite presque identique, à ceci près qu'un fragment 'accepté' peut passer à l'état 'suggéré' puis à l'état 'supprimé', tandis qu'un fragment 'importé' est supprimé directement. Egalement, le comportement d'un fragment à l'état 'créé' peut être quasiment le même que celui à l'état 'gelé'. Ainsi, nous ne décrirons pas l'état 'créé' ni l'état 'importé', mais seulement les états 'accepté' et 'gelé'.

Après qu'un fragment (fragment destination) eut été importé à partir d'un fragment source, les nouveaux sous-fragments qui par la suite sont ajoutés dans le fragment source sont implicitement à l'état 'suggéré' au sein du fragment destination. En mode administration ils sont présentés à l'utilisateur qui peut alors les faire passer à l'état 'accepté' ou à l'état 'supprimé'. Ils ne sont pas présentés à l'utilisateur en mode normal et il n'est pas nécessaire qu'ils soient mémorisés au sein du fragment destination. Par contre, les sous-fragment qu'on a fait passer à l'état 'supprimé' doivent alors être mémorisés (au sein du fragment destination), pour que le système puisse distinguer les sous-fragments (supprimés) qui ne doivent pas être re-suggérés (en mode administration).

Dans le cas où, au sein d'un fragment destination qui est à l'état 'importé' ou 'accepté', tous les sous-fragments (correspondant à tous les descendants du fragment source correspondant au fragment destination) sont à l'état 'accepté', il n'est pas nécessaire de les mémoriser au sein du fragment destination ; un attribut « all-accepted="yes" » associé au fragment destination suffit. Les sous-fragments devront être mémorisés dans toute la branche (de l'arbre XML) comprenant au moins un descendant qui n'est pas à l'état 'accepté'.

Enfin, l'état 'gelé' d'un fragment destination correspond à une copie du fragment source correspondant.

L'application apte à gérer le mode administration

La page aspx reçoit le mode en tant que paramètre (lors de son appel) et le retransmet aux composants FgmtCtrl (en tant que propriété statique) :

```
FgmtCtrl.Mode = Request.Params["mode"];
```

Le programme aspx n'a pas beoin d'être modifiée.

Composants de présentation en mode administration

Les ajouts et modifications par rapport au code présenté à la section précédente sont en gras.

Les composants fait appel à une fonction « `AdminModeDraw` » apte à présenter à l'utilisateur des moyens pour manipuler les fragments (moyens qui incluent par exemple des poignées/cibles comme déjà décrit).

```
public class ctrlmode : Control
{
      bool _isfgmt=false;
      String element;
      Static String _mode="normal";
      String fgmt_id;

      public ctrlmode() {}
      public ctrlmode(String element)
      {
            this.element = element;
      }
      static public String Mode // received from FgmtCtrl
      {
            get { return _mode;}
            set { _mode = value; }
      }
      public String Element   // received from FgmtCtrl
      {
            get { return element;}
            set { element = value; }
      }
      public bool IsFgmt      // received from FgmtCtrl
      {
            get { return _isfgmt;}
            set { _isfgmt = value; }
      }
      public String Self      // received from FgmtCtrl
      {
            get { return fgmt_id;}
            set { fgmt_id = value; }
      }
      public void AdminModeDraw()
      {
            if (!_isfgmt) return;

            Label l = new Label();
```

```
            1.Text = element;
            switch(_mode)
            {
                  case "admin":
                        SomeButtons img = new SomeButtons();
                        Controls.Add(img);
                        break;
                  ...
            }
            Controls.Add(1);
      }
}
public class ctrl1 : ctrlmode
{
      Image _img;
      int _algn,_pos;

      public int Pos
      {
            get { return _pos;}
            set { _pos = value; }
      }
      public String Self
      {
            get { return _abspath;}
            set { _abspath = value; }
      }
      public String Image
      {
            get { return _img.ImageUrl; }
            set { _img.ImageUrl = value; }
      }
      public int Align
      {
            get { return _algn; }
            set { _algn = value; }
      }
      protected override void CreateChildControls()
      {
            AdminModeDraw();
            if ( _algn == 0)
                  _img.ImageAlign = ImageAlign.Left;
            else
                  _img.ImageAlign = ImageAlign.Right;
            Controls.Add(_img);
      }
}
public class recipient : ctrlmode
{
      public ArrayList _ctrls;
      public recipient() { _ctrls = new ArrayList(); }
      public void Add(Control obj)
      {
            _ctrls.Add(obj);
      }
}
public class ctrl4 : recipient
{
```

```
        int _pos;

        public ctrl4():base() { }
        public new void Add(Control obj)
        {
              base.Add(obj);
        }
        public int Pos
        {
              get { return _pos;}
              set { _pos = value; }
        }
        public String Self
        {
              get { return _abspath;}
              set { _abspath = value; }
        }
        protected override void CreateChildControls()
        {
              AdminModeDraw();
              Control obj;
              if (_ctrls.Count ==0 )
                    return;

              Table tbl;
              TableRow tr;
              TableCell td;

              tbl = new Table();
              tr = new TableRow();
              td = new TableCell();
              obj = (Control) _ctrls[0];
              td.Controls.Add(obj);
              tr.Cells.Add(td);
              tbl.Rows.Add(tr);
              tr = new TableRow();
              td = new TableCell();
              obj = (Control) _ctrls[1];
              ((ctrl1) obj).Align = _pos%2;
              td.Controls.Add(obj);
              obj = (Control) _ctrls[2];
              td.Controls.Add(obj);
              tr.Cells.Add(td);
              tbl.Rows.Add(tr);
              Controls.Add(tbl);
        }
}
```

Chapitre 2 – Architecture de vues éditables

Les principes de base décrits au chapitre précédent restent valables, cependant on va maintenant décrire et illustrer une méthode selon laquelle tous les éléments XML peuvent être des fragments selon l'usage qu'on en fait. En effet, contrairement au chapitre précédent où les fragments étaient spécifiés de manière permanente dans les données, ce sont maintenant les programmes (dans leur mode administration) qui décident quels éléments sont utilisés en tant que fragments.

Ainsi chaque fragment n'existe que pendant la durée d'exécution de la page (en mode administration) qui l'utilise.

Chaque page (programme aspx ou analogue) est assimilée à une « vue » (au sens des bases de données) sur les éléments XML qu'elle utilise. En mode administration[7] ce sont ces vues qui distinguent les nœuds fragments des autres nœuds XML et permettent de les manipuler, notamment de les éditer ou encore de les importer d'un endroit à l'autre dans un même document XML ou entre plusieurs documents dans un même ordinateur ou même entre des serveurs différents (par exemple via un service web chargé de gérer chaque document). Ainsi ce sont des vues éditables.

Attribut « Mode="admin" » des composants de présentation

Prenons un exemple de données XML très proche de celui présenté au chapitre précédent.

Document1'

```
<?xml version='1.0' ?>
<PAGE>
   <BLOC>
     <HEADBLOC>Ceci est un premier sous-titre</HEADBLOC>
     <SMALLBLOC src="image1.jpg" video="video1.mpeg"/>
     <BIGBLOC>texte texte texte texte texte texte </BIGBLOC>
   </BLOC>
   <BLOC>
     <HEADBLOC>Ceci est un autre sous-titre</HEADBLOC>
     <SMALLBLOC src="image2.jpg"/>
     <BIGBLOC>autre texte autre texte autre texte </BIGBLOC>
   </BLOC>
</PAGE>
```

Chaque composant FgmtCtrl a comme attribut la liste des modes administration pour lesquels l'élément auquel il s'applique (désigné par l'attribut **Element**) est spécifié comme étant un fragment. Dans le programme aspx ci-dessous, seul les éléments BLOC et HEADBLOC est défini comme étant fragment manipulable (dans un mode administration **Mode="admin"** ; noter que dans le cas général, plusieurs valeurs peuvent être données pour cet attribut).

aspx1.aspx

```
<%@ Register TagPrefix="App" NameSpace="WebControlLibrary123" Assembly="tagcontrols" %>
<%@ Page Language="C#" Inherits="Lib.PageCtrl" %>
<html>
<body>
<form runat="server">
```

[7] (on peut distinguer plusieurs modes administration et les nommer différemment)

```
<App:PageCtrl Data="..\document1.xml">

 <App:FgmtCtrl Element="PAGE"  Control="ctrl6" NameSpace="Lib" Asm="PWebControls" Args="Empty"
       runat="server">

  <App:FgmtCtrl Element="BLOC" Mode="admin" Control="ctrl4" NameSpace="Lib" Asm="PWebControls"
       Args="Empty" runat="server">

    <App:PropertyMapping Name="Align" Value="Const:1" Asm="PWebControls" runat="server"/>

    <App:FgmtCtrl Element="HEADBLOC" Mode="admin" Control="ctrl3" NameSpace="Lib" Asm="PWebControls"
       Args="Content"  runat="server"/>

    <App:FgmtCtrl Element="SMALLBLOC" Control="ctrl11" Target="aspx3.aspx?SMALLBLOC" NameSpace="Lib"
       Asm="PWebControls" Args="Content" runat="server">

       <App:PropertyMapping Name="Image" Value="@src" Asm="PWebControls" runat="server"/>

    </App:FgmtCtrl>

    <App:FgmtCtrl Element="BIGBLOC" Control="ctrl21" Target="aspx2.aspx?BLOC" NameSpace="Lib"
       Asm="PWebControls" Args="Content" runat="server" />

  </App:FgmtCtrl>

 </App:FgmtCtrl>

</App:PageCtrl>
</form>
</body>
</html>
```

Le programme aspx1.aspx ci-dessus présente la page esquissée à la figure 4.a et présente la même page en mode administration comme illustré schématiquement à la figure 4.b.

Les composants ctrl1 et ctrl2 ont un comportement particulier :

- ctrl2 présente à l'utilisateur un extrait du contenu textuel de l'élément BIGBLOC ainsi qu'un *lien hypertexte* dont l'activation déclenche l'appel de la page aspx2 (voir ci-après) qui présente BIGBLOC dans son intégralité mais ignore l'élément SMALLBLOC ;
- ctrl1 se présente sous la forme d'une image (celle définie dans l'élément SMALLBLOC : respectivement image1.jpg ou image2.jpg) munie d'un *lien hypertexte* dont l'activation appelle la page aspx3 (voir ci-après) dont le rôle est de présenter le film spécifié (le cas échéant) par l'attribut « video » de SMALLBLOC (« video1.mpeg » dans le premier BLOC ; pas d'attribut vidéo dans le second BLOC).

Le composant ctr21 a l'allure suivante :

```
public class ctrl21 : ctrlmode
{
      String _mesg,_link,_absolutepath;
      int _pos;

      public ctrl2() { _mesg = "body"; }
      public ctrl2(String msg) { _mesg = msg; }
      public int Pos
      {
            get { return _pos;}
            set { _pos = value; }
      }
      public String Target
      {
```

```
                get { return _link;}
                set { _link = value; }
        }
        public String Self
        {
                get { return _absolutepath;}
                set { _absolutepath = value; }
        }
        String getExtract()
        {
                ...
        }
        protected override void CreateChildControls()
        {
                AdminModeDraw();
                Literal l = new Literal();
                l.Text = "<div> <paragraph>"+ getExtract() +
                            " </paragraph> </div>" +
                            "<a href=\"+ _link + "="+ _absolutepath+
                            "\"> ... </a>";
                Controls.Add(l);
        }

}
```

aspx2.aspx

```
<%@ Register TagPrefix="App" NameSpace="WebControlLibrary123" Assembly="tagcontrols" %>
<%@ Page Language="C#" Inherits="Lib.PageCtrl" %>
<html>
<script language="C#" runat="server">
        void Page_Load()
        {
                FgmtCtrl2.Element = Request.Params["BLOC"];
        }
</script>

<body>
<form runat="server">

<App:PageCtrl Data="..\document1.xml">

  <App:FgmtCtrl Element="PAGE" Control="ctrl6" NameSpace="Lib" Asm="PWebControls" Args="Empty"
        runat="server">

   <App:FgmtCtrl ID="FgmtCtrl2" Control="ctrl4" Mode="admin" NameSpace="Lib" Asm="PWebControls"
        Args="Empty" runat="server">

     <App:PropertyMapping Name="Align" Value="Const:1" Asm="PWebControls" runat="server"/>

     <App:FgmtCtrl Element="HEADBLOC" Control="ctrl3" NameSpace="Lib" Asm="PWebControls" Args="Content"
        runat="server"/>

     <App:FgmtCtrl Element="BIGBLOC" Control="ctrl21" NameSpace="Lib" Asm="PWebControls" Args="Content"
        runat="server" />

   </App:FgmtCtrl>

  </App:FgmtCtrl>

</App:PageCtrl>
</form>
</body>
</html>
```

L'élément BLOC est éditable, cependant les modifications effectuées seront directement visibles dans la page aspx1.aspx. Nous verrons plus loin comment importer un fragment et le modifier indépendamment de la source.

En variante, ce programme peut aussi être le suivant (dans ce cas dans aspx1.aspx la valeur de l'attribut Target doit être "aspx2.aspx?BIGBLOC"):

aspx2'.aspx

```
<%@ Register TagPrefix="App" NameSpace="WebControlLibrary123" Assembly="tagcontrols" %>
<%@ Page Language="C#" Inherits="Lib.PageCtrl" %>
<html>
<script language="C#" runat="server">
        void Page_Load()
        {
                FgmtCtrl2.Element = Request.Params["BIGBLOC"];
        }
</script>

<body>
<form runat="server">

<App:PageCtrl Data="..\document1.xml">

    <App:FgmtCtrl ID="FgmtCtrl2" Control="ctrl2" Mode="admin" NameSpace="Lib" Asm="PWebControls"
        Args="Content" runat="server" />

</App:PageCtrl>
</form>
</body>
</html>
```

Dans ce dernier cas on peut éditer BIGBLOC dans la page affichée en mode administration car le composant **"FgmtCtrl2"** est en **Mode="admin"**.

Le composant ctrl11, qui renseigne l'attribut Target du composant en question dans le programme appelant (aspx1.aspx), a l'allure suivante :

```
public class ctrl11 : ctrlmode
{
        Image _img;
        int _algn,_pos;
        String _link;

        public int Pos
        {
                get { return _pos;}
                set { _pos = value; }
        }
        public String Target
        {
                get { return _link;}
                set { _link = value; }
        }
        public String Self
        {
                get { return _abspath;}
                set { _abspath = value; }
        }
        public String Image
        {
                get { return _img.ImageUrl; }
                set { _img.ImageUrl = value; }
```

```
        }
        public int Align
        {
              get { return _algn; }
              set { _algn = value; }
        }
        protected override void CreateChildControls()
        {
              AdminModeDraw();
              Literal l=new Literal();
              l.Text = "<a href=\"" + _link + "="_abspath +
                          "\"><img src=\"+_imgUrl+"\> </a>";
              Controls.Add(l);
        }
    }
```

Enfin voici l'allure de la page aspx qui présente la vidéo dans les cas où elle est spécifiée dans le fragment SMALLBLOC en question:

aspx3.aspx

```
<%@ Register TagPrefix="App" NameSpace="WebControlLibrary123" Assembly="tagcontrols" %>
<%@ Page Language="C#" Inherits="Lib.PageCtrl" %>
<html>
<script language="C#" runat="server">
      void Page_Load()
      {
            FgmtCtrl1.Element = Request.Params["SMALLBLOC"];
      }
</script>
<body>
<form runat="server">

<App:PageCtrl Data="..\document1.xml">

    <App:FgmtCtrl ID="FgmtCtrl1" Element="SMALLBLOC" Control="ctrlMpeg" Mode="admin"
       NameSpace="Lib" Asm="PWebControls" Args="Content" runat="server">

</App:PageCtrl>
</form>
</body>
</html>
```

Le composant ctrlMpeg présente le film si l'attribut video est renseigné, sinon il présente l'image donné dans l'attribut src de l'élément courant qui lui est passé dans le constructeur. Dans les deux cas, des boutons d'administration seront ajoutés en **Mode="admin"**.

```
public class ctrlMpeg : ctrlmode
{
      XmlNode elmt;
      int _pos;

      Public ctrlMpeg(XmlNode elem)
      {
            elmt = elem;
            ...
      }
      public int Pos
      {
            get { return _pos;}
            set { _pos = value; }
      }
      public String Self
```

```
        {
              get { return _abspath;}
              set { _abspath = value; }
        }
        protected override void CreateChildControls()
        {
              AdminModeDraw();
              If (video attribute exists for elmt)
                    Controls.Add("...video player...");
              else if (src attribute exists for elmt) {
                    Literal l =new Literal();
                    l.Text = "<img src=\"" + elmt.Attribute("src")
                                +"\"/>"
                    Controls.Add(l);
              }
              else
                    Controls.Add("...");
        }
  }
```

Fragments importés

On va maintenant décrire un exemple où on va importer des fragments (et donc mémoriser leurs références vers leurs sources).

Reprenons l'exemple de données XML.

Document1

```
<?xml version='1.0' ?>
<PAGE>
   <BLOC>
     <HEADBLOC>Ceci est un premier sous-titre</HEADBLOC>
     <SMALLBLOC src="image1.jpg" video="video1.mpeg"/>
     <BIGBLOC>texte texte texte texte texte texte </BIGBLOC>
   </BLOC>
   <BLOC>
     <HEADBLOC>Ceci est un autre sous-titre</HEADBLOC>
     <SMALLBLOC src="image2.jpg"/>
     <BIGBLOC>autre texte autre texte autre texte </BIGBLOC>
   </BLOC>
</PAGE>
```

On va maintenant importer un fragment BLOC de la même manière que dans le chapitre précédent. Ceci peut se faire en présentant les deux documents dans des pages en mode administration.

Document2 (webservice123) :

```
<?xml version='1.0' ?>
<PAGE >
 <SOMETHING>
   <BLOC>
     <HEADBLOC >Sous-titre du bloc externe</HEADBLOC>
     <SMALLBLOC src="image3.jpg"/>
   </BLOC>
```

```
 </SOMETHING>
</PAGE>
```

Document1 (après l'importation)

```
<?xml version='1.0' ?>
<PAGE>
   <BLOC>
     <HEADBLOC>Ceci est un premier sous-titre</HEADBLOC>
     <SMALLBLOC  src="image1.jpg"  video="video1.mpeg"/>
     <BIGBLOC>texte texte texte texte texte texte </BIGBLOC>
   </BLOC>
   <BLOC state="imported" fragment-ref="webservice123/GetFragment?Path=BLOC[1]">
     <HEADBLOC state="accepted"
       fragment-ref="webservice123/GetFragment?Path= BLOC[1]/HEADBLOC"/>
     < SMALLBLOC state="suppressed"
       fragment-ref ="webservice123/GetFragment? Path= BLOC[1]/SMALLBLOC"/>
   </BLOC>
   <BLOC>
     <HEADBLOC>Ceci est un autre sous-titre</HEADBLOC>
     <SMALLBLOC src="image2.jpg"/>
     <BIGBLOC>autre texte autre texte autre texte </BIGBLOC>
   </BLOC>
</PAGE>
```

Noter que l'attribut state est à « state="created" » par défaut.

Le document 1 étant maintenant modifié, les trois programmes aspx incluront automatiquement le nouveau BLOC importé dans leur propre présentation.

Chapitre 3 – Indexation et sélection de fragments

L'objet de ce chapitre est de décrire le procédé d'indexation et sélection de fragments (ou autres entités d'informations) selon le premier objet de l'invention.

Bien que le procédé permette, de manière générale, de sélectionner une entité d'informations parmi un ensemble d'entité d'informations supplémentaires, on va maintenant décrire son application sur l'architecture présentée au chapitre précédent pour sélectionner un fragment parmi un ensemble de fragments supplémentaires. L'homme du métier saura adapter le procédé à d'autres applications.

Description du procédé

Parmi un ensemble de « fragments supplémentaires » donnés, on cherche à sélectionner ceux qui sont les plus pertinents par rapport à un « fragment de départ ».

Concrètement, un fragment destiné à être présenté à l'utilisateur peut être spécifié en désignant un fragment de départ ainsi qu'un ensemble de fragments supplémentaires, de manière à ce que le programme de présentation sélectionne automatiquement, pour ce fragment à présenter à l'utilisateur, le fragment supplémentaire le plus pertinent.

Pour ce faire ce programme applique 3 étapes (qui sont décrites ci-après) :

1. Sélection de pages Web les plus similaires au fragment de départ
2. Sélection des pages Web les plus pertinentes par rapport aux pages résultant de l'étape 1
3. Sélection des fragments supplémentaires les plus similaires aux pages résultant de l'étape 2.

Etape 1 : Sélection de pages Web les plus similaires au fragment de départ

On connaît déjà de nombreuses méthodes dans l'état de la technique pour rechercher sur la Toile des pages similaires ou proches à un document donné. On peut notamment extraire des mots-clé du fragment de départ puis soumettre ces mots-clé à un moteur de recherche qui retournera les pages recherchées. Diverses techniques d'extraction automatique de mots-clé d'un texte existent déjà dans l'état de la technique[8]. Les moteurs de recherche sur le Web sont également courants. On ne décrira donc pas cette étape d'avantage.

On présentera plus loin (voir la section « Procédé permettant (notamment) de rechercher des pages pertinentes ») un nouveau procédé automatique ou semi-automatique de recherche de pages pertinentes sur le Web, qui peut remplacer la présente étape et même d'ailleurs aussi l'étape suivante éventuellement.

Les adresses des pages Web les plus similaires ainsi trouvées permettront une première indexation du fragment de départ. En effet, dans la mesure où un fragment de départ fera partie de l'ensemble des fragments supplémentaires, il est avantageux de mémoriser les adresses (url) de ses pages similaires ou proches qui résultent de l'application de l'étape 1.

[8] Voir par exemple les travaux de Luhn et Edmunson, « KEA : Practical automatic keyphrase extraction. In ACM DL, pages 254-255, 1999 » ou encore la méthode GenEx de Peter D. Turney.

Etape 2 : Sélection des pages Web les plus pertinentes par rapport aux pages Web les plus similaires au fragment de départ

Ceci fait l'objet du chapitre suivant (« Calcul des scores de pertinence »). En effet, on y présente un procédé permettant de rechercher sur le Web les pages les plus pertinentes par rapport à un ensemble d'url (URI) de pages Web données (en l'occurrence, par rapport aux pages résultant de l'étape 1).

Etape 3 : Sélection des fragments supplémentaires les plus similaires aux pages Web les plus pertinentes par rapport aux pages Web les plus similaires au fragment de départ

En résultat de l'étape 2, on a maintenant un ensemble (de taille limitée) des pages Web les plus pertinentes (par rapport au fragment de départ).

Il s'agit maintenant de sélectionner, parmi les fragments supplémentaires, ceux qui sont les plus similaires ou proches de ces pages Web les plus pertinentes. Or on connaît dans l'état de la technique de nombreuses méthodes[9] pour sélectionner des textes similaires ou proches à des textes donnés. La section suivante en présente une nouvelle (qui représente ainsi un troisième objet de l'invention).

Dans le cas où on indexe les fragments supplémentaires, c'est-à-dire on leur associe de manière permanente les adresses des pages Web qui leurs sont similaires ou proches[10], la sélection peut être immédiate. En effet, on peut directement sélectionner les fragments supplémentaires dont les adresses des pages Web similaires ou proches ont une grande intersection avec les adresses des pages Web les plus pertinentes résultant de l'étape 2.

[9] Celles-ci incluent notamment les techniques TF/IDF (Term Frequency / Inverse Document Frequency) et LSI (Latent Semantic Indexing).

[10] Ceci revient à appliquer l'étape 1 sur les fragments supplémentaires (voir plus haut la description de l'étape 1).

Procédé permettant (notamment) de rechercher des fragments pertinents

L'idée : exploiter les liens implicites qui existent entre les phrases

Appelons « document textuel » le fragment de départ[11]. Tout d'abord on épure le document textuel en lui ôtant tous les mots « trop courants » (que l'on appelle « stop words » dans le jargon) tels que les articles et les pronoms. Il ne reste ainsi que les mots signifiants.

On décompose le document textuel ainsi épuré (et étendu le cas échéant, comme mentionné en note de bas de page) en unités de textes, telles que des phrases.

On va maintenant exploiter les liens implicites existant entre unités de texte, un tel lien existant entre deux unités quand elles ont un mot en commun.

Si (de préférence) on a à disposition au moins une unité de texte déjà sélectionnée comme étant déjà pertinente, on procède comme suit :

Pour tous les mots figurant dans l'unité (ou les unités) pertinente(s), on recherche les unités qui sont co-citées et on leur attribue un score de pertinence comme décrit à la section suivante (« Détermination des scores de pertinence des unités co-citées »). On peut ensuite attribuer un score aux mots ou groupes de mots contenus dans les unités co-cités en additionnant les scores de pertinences des unités dans lesquelles ils se trouvent (on peut les ramener à une échelle commune en les normalisant)[12].

Si au contraire on n'a pas à disposition au moins une unité de texte déjà sélectionnée comme étant pertinente, on procède comme décrit à la section « Détermination des scores de pertinence des unités de texte ». Egalement, on peut ensuite attribuer un score aux mots ou groupes de mots contenus dans les unités de texte en additionnant les scores de pertinence des unités de texte dans lesquelles ils se trouvent (et en normalisant).

Extraire les mots-clé ou sélectionner directement les documents supplémentaires

On peut alors soit directement utiliser les mots ou groupes de mots ayant les scores les plus élevés comme mots-clé constituant une requête à un moteur de recherche, soit étendre le procédé pour sélectionner les fragments supplémentaires les plus pertinents. Dans ce dernier cas on procède comme suit :

Ayant attribué un score de pertinence aux unités de texte (selon l'une des méthodes évoquées ci-dessus), on sélectionne celles qui ont les scores les plus élevés comme étant « déjà pertinent ». On applique alors le même procédé (décrit à la section « Détermination des scores de pertinence des unités co-citées ») pour attribuer des scores à des unités de texte de documents (fragments) supplémentaires (le procédé est appliqué sur l'ensemble des documents plutôt que sur un seul document). Ensuite on attribue un score de pertinence de document aux documents

---

[11] On peut optionnellement étendre le fragment de départ en lui concaténant des fragments ou documents qu'elle cite, notamment par des liens hypertextes. L'extension peut se faire à partir des liens hypertextes contenu dans le fragment de départ seulement ou, en plus, en suivant les liens hypertextes contenus dans les documents eux-mêmes cité, et ainsi de suite pour un nombre de niveaux donné ou selon d'autres conditions.

[12] Les mots ou groupes de mots peuvent éventuellement être complétés par les adjectifs qui s'y appliquent majoritairement.

supplémentaires (pris individuellement) en prenant la moyenne des scores de pertinence des unités qu'ils contiennent. On sélectionne les documents supplémentaires ayant les meilleurs scores de pertinence de documents.

Une autre application possible du procédé est aussi à noter: les unités de texte ayant les meilleurs scores peuvent être assemblés pour former un résumé.

Détermination des scores de pertinence des unités co-citées

On identifie les unités de texte comprenant au moins un mot en commun avec l'unité (ou l'ensemble des unités) pertinente(s), pour former un groupe d'unités de texte citantes. On crée (temporairement) un lien à partir de chaque unité de texte citante vers l'unité (ou l'ensemble des unités) de texte pertinente(s).

On identifie les unités de texte contenant au moins un mot également contenu dans les unités de texte citantes, pour former un groupe d'unités de texte co-citées. On crée (temporairement) un lien à partir de chaque unité citante vers chaque unité co-citée avec laquelle ladite unité citante possède au moins un mot en commun.

On applique ensuite les procédés de calcul de scores de pertinence décrits au chapitre suivant. L'ensemble des identifiants des unités de texte pertinentes constitue les URI de la requête. L'ensemble des identifiants des unités de texte citantes constitue l'ensemble R^-. L'ensemble des identifiants des unités de texte co-citées constitue l'ensemble des « pages candidates »[13], c'est-à-dire l'ensemble R^+. Et ainsi de suite... Le procédé symétrique (voir la section « Cadre général pour le traitement par l'aval ») est ici équivalent.

Détermination des scores de pertinence des unités de texte

Les unités de textes sont ordonnées selon leur emplacement dans le texte. A chaque unité de texte est attribué un nombre de liens entrants (à partir des unités précédentes) et un nombre de liens sortants (vers les unités suivantes).

Le nombre de liens entrants est le total des unités précédentes dans le texte qui ont un mot en commun. Soit E= nombre de liens entrants +1.

Le nombre de liens sortants est le total des unités suivantes dans le texte qui ont un mot en commun. Soit S = nombre de liens sortants +1.

On va appliquer un coefficient pour contrebalancer l'effet de la position de l'unité dans le texte sur son nombre de liens entrants et sortants. Pour un texte de n unités, pour l'unité de texte ayant la position m, le nombre de liens entrants sera pondéré par $1 - (m+1)/(n+1)$, et le nombre de liens sortants sera pondéré par $(m+1)/(n+1)$.

Le score appliqué à chaque unité sera tout simplement la somme des liens entrants et sortants pondérés : score = $E^*(1 - (m+1)/(n+1)) + S^*(m+1)/(n+1)$.

[13] Les « pages » du chapitre suivant sont ici des unités de texte.

Chapitre 4 – Calcul des scores de pertinence

Introduction

Les approches existantes de l'état courant de la technique permettent de trier ou sélectionner des pages selon un critère de qualité « dans l'absolu ». En effet, selon ces approches, une meilleure page est soit une page plus populaire soit une page qui cite plus de pages plus populaires. Le présent procédé permet, au contraire, de trier ou sélectionner des pages selon un critère de pertinence par rapport à un ensemble d'URI[14] donnés comme étant eux-mêmes pertinents.

Pour faciliter la lecture, on va considérer les 7 ensembles suivants (voir la figure 5):

- R est constitué par les pages de la requête (c'est-à-dire leurs URI) et on suppose ici que ces dernières sont homogènes au sens du procédé de base[15].
- R^- est l'ensemble des pages qui contiennent un lien vers[16] au moins une des pages de la requête.
- R^{-+} est l'ensemble des pages pointées (citées) par les pages R^-.
- R^{+-} est l'ensemble des pages qui citent les pages R^{-+} ($R^- \subset R^{-+-}$).
- R^+ est l'ensemble des pages citées par au moins une des pages de la requête (R).
- R^{+-} est l'ensemble des pages qui citent les pages R^+.
- R^{+-+} est l'ensemble des pages citées par les pages de R^{+-} ($R^+ \subset R^{+-+}$).

Homogénéité d'un ensemble de pages

On présentera un procédé de calcul de score de pertinence d'un URI par rapport à un ensemble d'URI donné en entrée (pour constituer ladite requête de recherche). Ce procédé est basé sur une analyse des liens hypertextes et ne nécessite pas d'analyser le contenu des pages pointées, ni le texte autour des liens hypertextes.

L'idée essentielle du calcul du score de pertinence (d'une page P_2 par rapport à une page donnée P_1) est la suivante[17] :

Soit p_1 la probabilité[18] qu'un auteur aléatoire (de page Web) mette dans une page un lien sur P_1.

Soit p_2 la probabilité qu'un auteur aléatoire mette dans une page un lien sur P_2.

Soit $p_{1\&2}$ la probabilité qu'un auteur aléatoire, mette dans une page un lien sur P_1 et un lien sur P_2.

$B(P_i)$ est l'ensemble des URIs des pages ayant un lien vers la page P_i.

---

[14] (ci-après appelés « requête » ou « requête de recherche ». Typiquement, les URIs figurant dans les bookmarks, ou liens favoris, peuvent constituer une telle requête.)

[15] On verra plus loin comment décomposer une requête en sous-requêtes homogènes.

[16] (autrement dit « qui citent », ou encore « qui pointent »)

[17] Ci-après, nous allons considérer que P_1 et P_2, (ou P_i, P_j, etc) sont des pages Web, bien que les procédés décrits soient bien plus généraux, comme on l'a déjà mentionné brièvement.

[18] La probabilité d'être intéressé par une (ou certaines) page(s) est approchée en comptant le nombre de pages qui ont un lien sur elle(s) et en divisant ce nombre par une estimation du nombre de pages qui auraient pu en avoir.

$F(P_i)$ est l'ensemble des URIs des pages vers lesquelles P_i a un lien.

La pertinence d'une page par rapport à un ensemble de pages peut être définie par la « quantité de raisons communes » d'être intéressé par toutes ces pages.

Des calculs algébriques permettent d'obtenir des équations donnant la quantité de raisons communes entre plusieurs pages. Cette quantité (ou proximité, ou encore homogénéité) est notée x, avec en indice les pages dont il est question ; la probabilité d'être lié à une certaine page P_i est notée p_i ; la probabilité d'être lié à *au moins* une page parmi P_i, P_j, ..., P_n est notée $p_{ij...n}$:

$\overline{x_{ij}} = \frac{\overline{p_i} \cdot \overline{p_j}}{p_\emptyset \cdot p_{ij}}$, $\overline{x_{ijk}} = \frac{\overline{p_i} \cdot \overline{p_j} \cdot \overline{p_k} \cdot \overline{p_{ijk}}}{p_\emptyset \cdot p_{ij} \cdot p_{ik} \cdot p_{jk}}$, et ainsi de suite (tous les sous-ensembles de taille impaire au numérateur, et les autres au dénominateur)[19].

Cette équation peut être notée de façon plus compacte ainsi : $\overline{x_S} = \prod_{P \subset S} \overline{p_P}^{\sigma_P}$ avec $\sigma_P = (-1)^{|P|}$.

L'approche de l'algorithme

Les probabilités dont il est question ci-dessus font intervenir le nombre (le comptage) des pages de R^- qui contiennent un lien donné ou un lien parmi un ensemble d'URI donnés (vers des pages de R^{-+}). On gagnerait à pondérer ce nombre par la *qualité de citation* (score hub, décrit plus loin) de chaque page qui contient un tel lien.

On voudrait ainsi qu'une page de R^- citant plus de meilleures pages (de R^{-+}) soit considérée comme étant de meilleure qualité de citation, et qu'en retour un poids plus fort lui soit donné dans le cadre du calcul des scores[20] des pages qu'elle cite (R^{-+}), les scores des pages de R^- et ceux des pages de R^{-+} s'influençant mutuellement dans une approche itérative (de renforcement bipartite) qui converge[21].

Le nombre de pages de R^{+-} citant chaque page candidate (de R^{-+}) intervient aussi dans les calculs. Or leur prise en compte coûte cher. On va alors approximer les résultats en ne considérant que celles qui citent les pages candidates ayant un bon score, ce score étant calculé d'abord en ne considérant que R^- et ensuite en étendant cet ensemble vers R^{+-} progressivement.

Trouver les pages récentes

Pour calculer le score de pertinence d'une page candidate, au lieu de prendre le résultat de l'équation de quantité de raisons directement, il est préférable

- de la prendre avec les cardinalités d'ensemble remplacées par le total des scores hub des pages en question et

[19] Les barres supérieures indiquent des compléments, et $p_\emptyset$, la probabilité d'aimer au moins une page d'un ensemble vide, est une constante égale à zéro ; elle est présente dans l'équation pour des raisons de cohérence.

[20] Rappelons qu'il s'agit ici de scores de pertinence par rapport à la requête, contrairement de l'état de la technique qui permet de déterminer un score de qualité « dans l'absolu ».

[21] Noter que le calcul du score de pertinence d'une page de R-+ peut résulter en une valeur négative (que l'on va alors neutraliser ; ceci est décrit plus loin). En effet, certaines pages peuvent être, non seulement pas proches de la requête, mais même antagonistes par rapport à elle (le fait d'y être intéressé diminue les chances d'aimer les pages de la requête et inversément).

- de multiplier ce résultat par le score autorité de la page candidate (simplement calculé à partir du total des scores hub des pages citantes), afin d'affaiblir ainsi les pages qui sont relativement moins fiables (car moins populaires).

Les scores hub des pages citantes (R^+) ainsi trouvés vont ensuite servir à déterminer les pages qui sont pertinentes sans être populaires, au moyen d'une équation de proximité, telle que $x_S = \frac{\left|\bigcap_{P_i \in S} B(P_i)\right|}{\left|\bigcup_{P_i \in S} B(P_i)\right|}$ dans laquelle les cardinalités d'ensemble (représentées entre barres verticales) vont être remplacées par le total des scores hub des pages en question[22].

En résumé : d'abord on détermine les pages pertinentes (et populaires à la fois) par la première méthode qui en plus donne les scores hub des pages citantes (et de manière fiable). Ces scores permettent ensuite de trouver les pages pertinentes mais pas (encore) populaires.

Régions de pertinence dans les pages

Après une première itération, dans les pages citantes le système peut

- repérer les régions contenant des liens dirigés sur des pages de R^+ ayant un bon score
- et commencer déjà à élaguer les liens qui ne sont pas situés dans ces régions.

Comme les liens en question se trouvent placés sous des nœuds d'une structure typiquement arborescente de document (tel qu'en HTML notamment), pour déterminer une région de pertinence il suffit de prendre les nœuds (minimaux) qui englobent tous les bons liens et de leur retrancher les sous-nœuds (maximaux) qui contiennent un mauvais lien et qui ne contiennent pas de bon lien. Par « bon » lien on entend : un lien dirigé sur une page ayant un bon score. Par « mauvais » lien, on entend : un lien qui a été explicitement refusé par l'utilisateur.

Calcul de pertinence par l'amont

L'algorithme permet, ayant un ensemble homogène (ayant une homogénéité suffisante) d'URIs associé à des pages proches (un « noyau »), d'obtenir une liste d'URIs de pages qui sont intéressantes relativement à cet ensemble. Il sera décrit plus loin comment exploiter cet algorithme pour obtenir un ensemble de pages intéressantes pour un ensemble inhomogène que le concepteur d'un récipient aura mis ensemble.

En entrée, cet algorithme prend

- un ensemble K d'URIs de référence (« Kernel »)
- un ensemble A d'URIs candidats (« Authority »)
- un ensemble H d'URIs candidats pivots (« Hub »)
- un ensemble T d'URIs à refuser (« Trash »)

[22] On peut dire que l'on remplace les cardinalités par des « cardinalités pondérés », les poids étant les scores hub.

On a : $K^- \subset H \subset A^-$ et $T \cap K = \emptyset$. (E étant un ensemble d'URIs, $E^- = \bigcup_{P_i \in E} B(P_i)$ et $E^+ = \bigcup_{P_i \in E} F(P_i)$)

1. Associer à chaque page P_i de H, un nombre h_i, mis initialement à $\frac{1}{|H|}$, son score hub[23].
2. (Re-)calculer les scores autorité :
   a. Pour chaque page P_i de A, en commençant par celles de K, associer un nombre a_i, son score autorité, égal à $\sum_j l_{ji} \cdot h_j$, où $l_{ji} = \begin{cases} 0 \text{ s'il n'y a pas de lien entre } P_j \text{ et } P_i \\ 1 \text{ s'il y a un lien entre } P_j \text{ et } P_i \end{cases}$.
   b. Une optimisation possible mais dangereuse : si, pour certaines pages, a_i est suffisamment proche de sa valeur calculée précédemment (le cas échéant), et que les scores autorité des pages de K n'ont pas varié non plus, nous pouvons garder l'ancienne valeur de r_i pour cette page, pour économiser les calculs.
3. (Re-)calculer les scores de pertinence :
   a. Pour chaque page P_i de A calculer r_i^+, égal à $w_{i \cup K}$

   $r_i^+ = w_{i \cup K}$

   et dans le cas où le résultat est négatif (cas d'une page antagoniste à R) neutraliser les liens entrants de manière à avoir $r_i^+ = 0$.

   L'homogénéité par l'amont w_S d'un ensemble S est définie comme suit:

   $$\overline{w_S} = \prod_{P \subset S} \overline{a_P}^{\sigma_P}, \text{ où}$$

   - $\sigma_P = \begin{cases} -1 \text{ si P contient un nombre pair de pages} \\ +1 \text{ sinon} \end{cases}$
   - $a_P = \Delta \sum_j h_j l_{jP}$ où

     Δ est une constante arbitraire inférieure mais proche de 1 (elle sert à éviter des divisions par zéro mais ne change pas le principe de l'algorithme. Si l'ensemble H est plus grand que K^- alors cette constante peut être égale à un

     $$l_{jP} = \begin{cases} +1 \text{ si } \exists P_i \in P \mid l_{ji} = +1 \\ 0 \text{ sinon} \end{cases},$$

     avec $l_{ji} = \begin{cases} 0 \text{ s'il n'y a pas de lien entre } P_j \text{ et } P_i \\ 1 \text{ s'il y a un lien entre } P_j \text{ et } P_i \end{cases}$

     En d'autres termes, l_{jP} est égal à 1 s'il y a un lien
     - d'une page P_j (de H)
     - à au moins une page P_i de P

     et zéro sinon.

[23] Ainsi, avantageusement, la somme des |H| scores h_i est égale à 1.

Ceci signifie tout simplement que a_P est le total des scores hub des pages (de *H*) qui pointent sur au moins une page de *P* (*P* étant le sous-ensemble courant de *S* qui est considéré).

Pour chaque lien l_{ji} existant on peut lui associer un poids en fonction de la proximité des pages P_i et P_j et améliorer ainsi le résultat - voir plus loin « Filtrer le Népotisme ».

Ici, puisque $\forall P_i \in K$ on a $r_i^+ = w_K$ (la pertinence est la même pour toutes les pages P_i de *K*), le score de pertinence r_i^+ ne doit être calculée qu'une seule fois pour les pages de *K* (elle sera d'ailleurs déjà calculée lors de la procédure de découpage de la requête *R* en sous-requêtes (noyaux) *K*, et sera donc déjà connue à l'entrée de la procédure).

b. (Ce point sera sauté la première fois.) Pour avoir leur somme égal à 1, on doit diviser chaque r_i^+ par la somme $\sum|r_i^+|$ de toutes les valeurs absolues des r_i^+. Soit $\delta = \sum_i \left| r_i - \frac{r_i^+}{\sum|r_i^+|} \right|$, la variation globale du score de pertinence.

Si $\delta<\varepsilon$ ($\varepsilon>0$ étant une marge d'erreur), on considère avoir convergé et le procédé s'arrête. Sinon, le procédé continue.

c. On remplace r_i par $\frac{r_i^+}{\sum_i|r_i^+|}$

$$r_i \mapsto \frac{r_i^+}{\sum_i|r_i^+|}$$

on peut aussi utiliser un facteur de frottement τ :

$r_i \mapsto \tau r_i + \overline{\tau}\frac{r_i^+}{\sum_i|r_i^+|}$. ($\tau \in [0;1[$, on prendra de préférence une valeur très petite e.g. 0.01 pour que dans les cas où ce n'est pas nécessaire le nombre d'itérations ne change pas)

4. [24]Pour chaque page P_i de *H* :
   a. Trouver tous les liens qui pointent sur une page ayant un score de pertinence plus grand qu'un seuil epsilon à choisir ($\varepsilon>0$).
   b. Trouver I_i, le plus petit élément HTML [25] contenant la totalité des liens trouvés au point a ci-dessus.
   c. Pour chaque lien pointant sur une page de *T* (si *T* n'est pas vide), trouver le plus grand élément HTML le contenant (s'il y en a) et ne contenant pas de lien trouvé au point a. ci-dessus, et l'enlever de I_i.

[24] Ce point peut éventuellement être ignoré après la première fois.
[25] (ou autre représentation analogue...)

d. On garde tous les liens restant dans I_i et on supprime les autres (ou bien on les neutralise en mettant leur l_{ij} à zéro)

5. Recalculer les scores hub:
   a. Pour chaque page P_i de H, calculer $h_i^+ = \sum_j l_{ij} r_j$, la somme des scores de pertinence des pages pointées.
   b. $h_i \mapsto \frac{h_i^+}{\sum \left| h_i^+ \right|}$

      (La division par $\sum \left| h_i^+ \right|$ est, comme pour le score de pertinence, pour garder leur somme égale à 1).

Ensuite retourner au point 2.

Extension R^{+-}

Initialement, pour ne traiter qu'un nombre réduit de pages, les scores de pertinence peuvent être calculés sur la base de R (si on avait pris $H=R$). Ceci ne sera alors qu'une approximation. En effet, pour que les scores soient corrects, il faudrait les calculer en se basant plutôt sur $H=R^{+-}$. Mais comme la constitution de R^{+-} est relativement coûteuse, on ne prendra qu'un sous-ensemble : on prendra pour R^{+-} seulement les pages pointant sur les pages de A qui ont un bon score.

Ainsi[26], on va ajouter une sous-étape avant la fin de l'étape 2.a :

2.a.1. Dans le cas où le score r_i^+ de la page courante (P_i de A) est suffisant[27], on recalcule r_i^+ après avoir inséré dans H les nouvelles pages de $B(P_i)$

$$H \mapsto B(P_i) \bigcup H .$$

Défavoriser les pages moins fiables

On introduit un score autorité pour les pages de A et l'équation r_i^+ est $r = w_{i \cup K} \cdot a_i$ (plutôt que $r = w_{i \cup K}$). Le nouveau coefficient a_i permettra d'affaiblir les pages peu fiables (par le fait qu'ils sont peu populaires). En outre, l'équation sera plus cohérente dans la mesure où le score pertinence ne sera plus le même pour toutes les pages de la requête.

La procédure est maintenant la suivante :

1. Ce point est le même que celui de l'algorithme de calcul de scores de pertinence présenté plus haut.
2. Ce point ne change pas non plus.
3. (Re-)calculer les scores de pertinence :
   a. Pour chaque page P_i de A calculer r_i^+, égal à $w_{i \cup K} \cdot a_i$ et dans le cas où le résultat est négatif (cas d'une page antagoniste à R) neutraliser les liens entrants de manière à avoir $r_i^+ = 0$.

[26] Plusieurs méthodes peuvent être utilisées ; nous présentons ici la préférée.

[27] (c'est-à-dire supérieur à un seuil choisi ; ce seuil pourra être fonction de la cardinalité courante de H, en effet plus on se rapproche de R^{+-} (e.g. H_{final}) plus le score calculé a des chances d'être déjà correct)

b. Poursuivre à partir du point 3.b de l'algorithme de calcul de scores de pertinence présenté précédemment.

Filtrer le Népotisme

Pour chaque lien l_{ji} existant on peut lui associer un poids en fonction de la proximité des pages P_i et P_j et améliorer ainsi le résultat. On appelle cela « filtrer le Népotisme » car cela permet de diminuer l'importance des liens entre pages qui se promeuvent mutuellement. Typiquement on arrive ainsi à filtrer par exemple les liens des « sommaires » et autres « menus » qui, de manière répétitive, se trouvent dans toutes les pages d'un site.

L'idée de base consiste à affaiblir les liens reliant deux pages que nous savons proches, en affectant un poids à chaque lien, poids qui sera égal au complément de la proximité des deux pages reliées (plus la proximité est grande, plus le lien doit être affaibli). Une fois que les liens sont ainsi pondérés, il est possible de calculer l'homogénéité d'un ensemble de pages en utilisant non plus le nombre de pages citantes, mais la somme de leurs poids.

Au point 3.a de l'algorithme, on remplace dans la définition de du score autorité $\sum_j h_j l_{jP}$ par

$\sum_j h_j \ell_{jP}$ où $\ell_{jP} = \min\left[1; \max_{P_i \in P}\left(l_{ji} \cdot \overline{x_{ji}}\right)\right]$

Explications :

- $l_{ji} \cdot \overline{x_{ji}}$ est le complément de la proximité entre la page P_j et la page P_i s'il y a un lien de la page P_j à la page P_i, et zéro sinon
- $\max_{P_i \in P}\left(l_{ji} \cdot \overline{x_{ji}}\right)$ est le complément de la proximité entre la page $P_j \in H$ en question et la page $P_i \in P$ pour laquelle le lien entre P_j et P_i présente la proximité minimum
- $\min\left[1; \max_{P_i \in P}\left(l_{ji} \cdot \overline{x_{ji}}\right)\right]$ signifie que cette valeur est tronquée supérieurement à 1
- et toujours $l_{ji} = \begin{cases} 0 \text{ s'il n'y a pas de lien entre } P_j \text{ et } P_i \\ 1 \text{ s'il y a un lien entre } P_j \text{ et } P_i \end{cases}$

En d'autres termes, s'il y a au moins un lien

- de la page P_j (de H) en question
- à une page P_i de P,

ℓ_{jP} est égal au complément de la proximité entre la page P_j et la page P_i qui lui est la moins proche et vers laquelle elle possède un lien. $\sum_j \ell_{jP}$ est la somme des poids ainsi associés aux pages de H qui pointent sur au moins une des pages du sous-ensemble P considéré.

Pour déterminer la proximité x_{ji}, on peut prendre l'équation de quantité de raisons communes (déjà décrite) : $\overline{x_{AB}} = \dfrac{\overline{p_A} \cdot \overline{p_B}}{p_\emptyset \cdot p_{AB}}$

La figure 6 présente un exemple où le nombre de pages pointant sur la page A est égal à 0.9+0.2+0.4+***0.8***=**2.3**

Le nombre de pages pointant sur la page B est égal à 0.9+***0.1***+0.3+0.5=**1.8**

Le nombre de pages pointant sur A ou B (N p_{AB}) est égal à 0.9+0.2+0.9+0.8+0.3+0.5=**3.6**
Ainsi, si on considère que $|H|+\hbar=100$, le calcul de la proximité de A et B donne :

$\overline{x_{AB}}=\frac{\overline{p_A}\cdot\overline{p_B}}{p_\emptyset\cdot p_{AB}}=\frac{0.977\cdot 0.932}{1\cdot 0.964}$, ce qui donne $\tilde{x}_{AB}=\frac{x_{AB}}{p_B}\approx 0.264=26.4\%$.

Le filtrage du népotisme décrit ci-dessus utilise un facteur de népotisme $\overline{x_{ji}}$. Puisque nous avons maintenant les scores[28] des pages citantes, nous pouvons éventuellement améliorer le procédé en prenant $\overline{x_{ji}\cdot h_j}$ comme facteur de népotisme (au lieu de $\overline{x_{ji}}$), où h_j est le score de la page citante. En effet, il faut d'autant plus affaiblir un lien népotiste (d'une page citante P_j à une page citée P_i) que le score de la page citante P_j est faible.

Décomposer une requête en noyaux (sous-requêtes homogènes)

L'utilisateur fournit au système un ensemble R de pages auxquelles il est intéressé et éventuellement un ensemble de pages R_X de pages qu'il ne veut explicitement pas ($R\cap R_X=\emptyset$). Le système va identifier au sein de R au moins un groupe de pages « homogène » et va lancer une sous-requête séparée sur ce ou chaque groupe. Ces groupes sont appellés « kernel » (ou noyau). Pour former la réponse à l'utilisateur on prendra ensuite une combinaison des scores obtenus.

Procédure :

1. Pour chaque page P_i de R, trouver $B(P_i)$, l'ensemble de pages citant P_i.
2. Trouver $R^- = \bigcup_{P_i\in R} B(P_i)$, l'ensemble de pages citant au moins une page de R.
3. Dans les pages de R qui ne sont pas encore dans un noyau (au début aucune ne l'est), trouver la page P_B ayant le plus grand ensemble $B(P_B)$ de liens entrants[29], et créer un noyau contenant seulement cette page. Ce noyau est maintenant K_C, le noyau courant en construction (à tout instant il n'y en a qu'un seul). Si toutes les pages se trouvaient dans au moins un noyau alors passer au point six.
4. Trouver les pages pertinentes par rapport à K_C (en utilisant l'algorithme de calcul de scores de pertinence) avec
   - $H=R^-$
   - $A=R$
   - $K=K_C$
   - $T=R_X$
5. Soit P_N la page de R, pas encore dans K_C, qui a le score de pertinence le plus élevé. Si son score de pertinence est inférieur à un score minimal fixé, retourner au point trois. (le noyau courant est maintenant complet). Sinon l'insérer dans K_C et repasser au point quatre. A noter qu'il ne sera pas nécessaire de réinitialiser les scores pivot et autorité, il est préférable de garder les dernières valeurs calculées, ainsi la convergence devrait être très rapide.

[28] (que ce soit de manière absolue ou par rapport à la requête)
[29] Dans le cas où on a les scores autorité des pages, ou autre score de popularité, on préfère se baser plutôt sur eux.

6. On a maintenant un ensemble de noyaux (sous-requêtes homogènes par l'amont) prêtes à être utilisées comme décrit dans ce document. Lorsqu'on veut calculer les scores de pertinence globalement à toute la requête on fait une moyenne arithmétique des résultats pour chacun des noyaux.

Variante

Au lieu de se baser sur l'équation d'homogénéité $\overline{x_S} = \prod_{P \subset S} \overline{p_P}^{(-1)^{|P|}}$ comme décrit jusqu'ici, le procédé de calcul de scores de pertinence peut être basé sur une autre équation d'homogénéité, comme par exemple $x_S = \frac{\left|\bigcap_{P_i \in S} B(P_i)\right|}{\left|\bigcup_{P_i \in S} B(P_i)\right|}$ ou encore $x_S = \frac{\left|\bigcap_{P_i \in S} B(P_i)\right|}{\left|\bigcup_{P_i \in S} B(P_i)\right|} \cdot \left(\frac{\overline{Min_{P_i \in S} |B(P_i)|}}{Max_{P_i \in S} |B(P_i)|}\right)$ dans lesquelles les cardinalités d'ensemble (représentées entre barres verticales) sont remplacées par le total des scores hub des pages en question[30].

Cadre général pour le traitement par l'aval

Au lieu de chercher les bonnes pages relativement à celles d'un noyau parmi les pages qui sont citées en commun avec elles il peut être intéressant d'effectuer les mêmes algorithmes dans l'autre sens, i.e. en cherchant parmi les pages qui citent les mêmes pages que le noyau, voire même d'effectuer les deux et de calculer une moyenne arithmétique.

Les algorithmes par l'aval sont identiques à ceux par l'amont sauf que B est remplacé par F et F est remplacé par B, et $^-$ est interverti avec $^+$ (eg. R^{-+} est remplacé par R^{+-}).

Attribution de pages « artificielles »

Les procédés par l'amont et par l'aval peuvent être avantageusement intégrés de la manière suivante : Après le traitement par l'amont (éventuellement même après chaque itération amont), aux pages candidates (R^{-+}) ayant obtenu un score de pertinence suffisant, on associe à l'aval un ensemble de pages supplémentaires (« pages artificielles ») dont la cardinalité est fonction dudit score de pertinence. Chaque page artificielle est aussi citée par (au moins) une page de la requête. On donne ainsi à l'aval un « avantage » aux scores de ces bonnes pages (de R^{-+}) trouvées par l'amont[31], et par conséquent on donne aussi indirectement un avantage aux scores des pages (de R^{+-+}) citées le cas échéant par ces bonnes pages.

Et réciproquement, après le traitement par l'aval (éventuellement même après chaque itération aval), on applique à l'amont le même procédé de manière symétrique. On favorise ainsi les bonnes pages de R^{+-} et indirectement les pages (de R^{-+-}) qui les citent le cas échéant.

Le fait de ne pas amalgamer les scores par l'amont (des pages R^{-+}) avec les scores par l'aval (pages R^{+-}) permet de les dissocier dans les calculs. Notamment, on peut diminuer l'influence des scores obtenus par l'aval dans les traitements par l'amont ou vice-versa.

---

[30] On peut dire que l'on remplace les cardinalités par des « cardinalités pondérés », les poids étant les scores hub.
[31] Noter que, avantageusement, ceci se fait sans amalgamer les scores de pertinence par l'amont et par l'aval.

Par ailleurs, grâce à cette idée de « pages artificielles », le présent procédé peut être appliquée en complément aux méthodes existantes dans l'état de la technique. En effet, une fois les scores obtenus pour chaque page, on peut modifier artificiellement les nombres respectifs des pages citantes et citées avant d'appliquer ces méthodes.

On peut arpenter (« crawling » en terminologie anglo-saxonne) le Web en suivant les liens (en amont et en aval) autour des pages des 7 ensembles précédemment citées, en tirant parti de l'ajout des pages artificielles pour avantager les pages Web liées aux pages qui sont plus pertinentes par rapport à la requête.

Arpenter le Web récursivement

Dans la mesure où les pages ayant les meilleurs scores sont censées être très pertinentes pour l'utilisateur (et dans la mesure où la pertinence est transitive), les procédés décrits ici pourront être récursivement appliqués sur ces dernières pour découvrir encore d'autres pages pertinentes. On peut ainsi arpenter le Web à partir de la requête de l'utilisateur.

La figure 7 présente de manière schématique un tel procédé : la recherche de pages pertinentes peut être appliquée récursivement en étendant la requête avec les « Bonnes pages trouvées par l'amont », « Bonnes pages trouvées par l'aval », « Bonnes pages hub » et « Bonnes pages autorités » qui dans la figure sont encadrés par des rectangles. A chaque récursion, les scores des meilleures pages trouvées deviennent un peu plus faibles (par le fait que les meilleures pages trouvées sont à chaque fois ajoutées dans la requête) et le procédé s'arrête quand les scores cessent d'être suffisants.

REVENDICATIONS

1. Procédé pour permettre l'accès par un utilisateur à des d'entités d'informations pertinentes à partir d'une entité d'informations de départ, chaque entité d'informations étant accessible par un identifiant (URI), caractérisé en ce qu'il comprend les étapes suivantes :

a) prévoir au moins une entité d'informations similaire, présentant un contenu similaire à celui de l'entité de départ, et déterminer l'identifiant de la ou de chaque entité d'informations similaire, et

b) déterminer à partir du ou de chaque identifiant d'entité d'informations similaire un ensemble d'un ou plusieurs identifiants d'entités d'informations pertinentes par rapport à la ou chaque entité d'informations similaire.

2. Procédé selon la revendication 1, caractérisé en ce qu'il comprend en outre l'étape suivante :

c) permettre à l'utilisateur l'accès à au moins certaines informations pertinentes à partir de leurs identifiants respectifs.

3. Procédé selon la revendication 1 ou 2, caractérisé en ce qu'il comprend en outre l'étape suivante :

d) à partir des identifiants d'entités d'informations pertinentes et d'un ensemble donné d'entités d'informations supplémentaires, sélectionner les entités supplémentaires les plus similaires aux entités d'informations pertinentes.

4. Procédé selon l'une des revendications 1 à 3, caractérisé en ce qu'il comprend une étape supplémentaire de tri des entités d'informations pertinentes par degré de pertinence.

5. Procédé selon la revendication 4, caractérisé en ce que l'étape de tri est précédée d'une étape de calcul d'un score de pertinence par rapport à la ou chaque entité d'informations similaires pour chacune des entités d'informations pertinentes.

6. Procédé selon l'une des revendications 1 à 5, caractérisé en ce que chaque entité d'informations est constituée par un fragment de page écrite en langage de marquage normalisé, ou par une telle page dans son ensemble.

7. Procédé selon la revendication 6, caractérisé en ce que chaque identifiant est constitué par un identificateur uniforme de ressource (URI) du fragment ou de la page.

8. Procédé selon l'une des revendications 1 à 7, caractérisé en ce que l'étape a) est réalisée par sélection par l'utilisateur d'une ou plusieurs entités d'informations similaires à l'entité d'informations de départ.

9. Procédé selon l'une des revendications 1 à 7, caractérisé en ce que l'étape a) est réalisée par mise en œuvre d'un processus de détermination automatique d'entités d'informations similaires.

10. Procédé selon l'une des revendications 1 à 7, caractérisé en ce que l'étape a) est réalisée par mise en œuvre d'un processus de détermination automatique d'entités d'informations

similaires, suivie d'une sélection par l'utilisateur d'une ou plusieurs entités d'informations similaires parmi les entités d'informations similaires déterminées par ledit processus.

11. Procédé selon l'une des revendications 1 à 10, caractérisé en ce que l'étape b) est réalisée par mise en œuvre d'un processus de détermination automatique d'entités d'informations pertinentes.

12. Procédé selon la revendication 11, caractérisé en ce que le processus de détermination automatique d'entités d'informations pertinentes comprend l'analyse d'une structure de graphe d'identifiants constituée par les identifiants d'entités d'informations et par les identifiants désignés par des liens activables par l'utilisateur contenus dans lesdites entités d'informations.

13. Procédé selon la revendication 12, caractérisé en ce que le processus de détermination automatique d'entités d'informations pertinentes comprend les opérations suivantes :

b1) identifier un ensemble d'entités d'informations citantes constitué par toutes les entités d'informations possédant un lien désignant l'identifiant de l'entité d'informations similaire ou d'au moins l'une des entités d'informations similaires,

b2) identifier un ensemble d'entités d'informations candidates constitué par l'ensemble des entités d'informations dont les identifiants sont désignés dans d'autres liens contenus dans les entités d'informations citantes,

b3) pour chaque entité d'informations candidate, calculer un score de pertinence d'entité d'informations candidate entre ladite entité d'informations candidate et l'entité d'informations similaire ou l'ensemble d'entités d'informations similaires, sur la base de l'existence, dans les entités d'informations citantes prises individuellement, à la fois d'un lien désignant l'identifiant de l'entité d'informations candidate et d'un lien désignant l'identifiant de l'entité d'informations similaire ou les entités d'informations similaires, et sur la base également de scores de pertinence d'entité d'informations citante affectés à chacune des entités d'informations citantes,

b4) pour chaque entité d'informations citante, recalculer un score de pertinence d'entité d'informations citante sur la base de l'existence, dans l'entité d'informations citante en question, de liens vers les entités d'informations candidates et sur la base également des scores de pertinence d'entité d'informations candidate attribuées aux entités d'informations candidates à l'étape b3),

b5) répéter le cas échéant l'étape b3) et le cas échéant une ou plusieurs fois l'étape b4) puis l'étape b3), pour toutes les entités d'informations candidates et citantes, et

b6) déterminer lesdites entités d'informations pertinentes comme étant les entités d'informations candidates qui présentent les meilleurs scores de pertinence d'entité d'informations candidate.

14. Procédé selon la revendication 12, caractérisé en ce que le processus de détermination automatique d'entités d'informations pertinentes comprend les opérations suivantes :

b1) identifier un ensemble d'entités d'informations citées constitué par toutes les entités d'informations identifiées dans des liens activables contenus dans l'entité d'informations similaire ou au moins l'une des entités d'informations similaires,

b2) former un ensemble d'entités d'informations candidates constitué par l'ensemble des autres entités d'informations ayant des liens activables identifiant lesdites entités citées,

b3) pour chaque entité d'informations candidate, calculer un score de pertinence d'entité d'informations candidate entre ladite entité d'informations candidate et l'entité d'informations similaire, ou l'ensemble d'entités d'informations similaires, sur la base de l'existence de liens

activables contenus dans l'entité d'informations candidate et dans le ou les entités d'informations similaire(s) et identifiant les entités d'informations citées, et sur la base également de scores de pertinence d'entité d'informations citée affectés à chacune des entités d'informations citées,

b4) pour chaque entité d'informations citée, recalculer un score de pertinence d'entité d'informations citée sur la base de l'existence, dans l'entité d'informations citée en question, de liens activables identifiant cette entité d'informations citée et contenus dans les entités d'informations candidates et sur la base également des scores de pertinence d'entité d'informations candidate attribuées aux entités d'informations candidates à l'étape b3),

b5) répéter le cas échéant l'étape b3) et le cas échéant une ou plusieurs fois l'étape b4) puis l'étape b3), pour toutes les entités d'informations candidates et citées, et

b6) déterminer lesdites entités d'informations pertinentes comme étant les entités d'informations candidates qui présentent les meilleurs scores de pertinence d'entité d'informations candidate.

15. Procédé selon la revendication 13, caractérisé en ce que le calcul de score de pertinence effectué à l'étape b3) comprend le calcul d'une pluralité de sommes de scores de pertinence d'entités d'informations citantes, chaque somme comprenant uniquement les scores de pertinences des entités d'informations citantes comprenant un lien vers une entité d'informations donnée constituée par l'entité d'informations candidate ou une entité d'informations similaire.

16. Procédé selon la revendication 15, caractérisé en ce qu'il comprend également le calcul d'au moins une somme de scores de pertinence d'entités d'informations citantes, chaque somme comprenant uniquement les scores de pertinence des entités d'informations citantes comprenant un lien vers l'une parmi un ensemble d'au moins deux entités d'informations données comprenant l'entité d'informations candidate et au moins une entité d'informations similaire.

17. Procédé selon l'une des revendications 1 à 16, caractérisé en ce que l'entité d'informations de départ est accessible par l'utilisateur et en ce que l'étape b) comprend la prise en compte d'informations spécifiques à l'utilisateur.

18. Procédé selon la revendication 17, caractérisé en ce que les informations spécifiques à l'utilisateur comprennent des identifiants d'entités d'informations mémorisés dans un historique de consultation d'entités d'informations par l'utilisateur.

19. Procédé selon la revendication 2 ou l'une des revendications 3 à 18 prise dans la dépendance de la revendication 2, caractérisé en ce qu'il comprend en outre une étape de présentation à l'utilisateur, de manière associée, d'au moins deux éléments choisis dans le groupe comprenant :

- une présentation de l'entité d'informations de départ
- un lien activable désignant l'identifiant de l'entité de départ ;
- des présentations des entités d'informations pertinentes déterminées à l'étape c) ;
- des liens activables désignant les identifiants des entités d'informations pertinentes déterminées à l'étape c).

20. Procédé selon la revendication 3 ou l'une des revendications 4 à 18 prise dans la dépendance de la revendication 3, caractérisé en ce qu'il comprend en outre la présentation à l'utilisateur, de manière associée, d'au moins deux éléments choisis dans le groupe comprenant :

- une présentation de l'entité d'informations de départ ;

- un lien activable désignant l'identifiant de l'entité d'informations de départ ;
- des présentations des entités d'informations supplémentaires les plus similaires déterminées à l'étape d) ;
- des liens activables utilisant les identifiants des entités d'informations supplémentaires les plus similaires déterminées à l'étape d).

21. Procédé de composition ou de modification d'un ensemble d'informations contenant la désignation d'une entité d'informations de départ donnée, caractérisé en ce qu'il comprend les étapes suivantes :
- mettre en œuvre le procédé selon la revendication 3 sur ladite entité de départ pour déterminer une ou plusieurs entités d'informations supplémentaires les plus similaires, et
- substituer à la désignation de l'entité de départ la désignation d'au moins l'une des entités supplémentaires les plus similaires.

22. Procédé selon la revendication 21, caractérisé en ce que l'ensemble d'informations possède une structure comprenant un arbre avec au moins un nœud au niveau duquel la désignation d'une entité d'informations de départ est effectuée, et en ce qu'une mise en œuvre du procédé selon la revendication 3 au niveau de ce nœud comprend à l'étape b) la prise en compte d'identifiants d'entités d'informations pertinentes associés à au moins un nœud ancêtre de ce nœud dans l'arbre.

23. Procédé d'exécution variable d'un programme d'affichage de données comportant des opérations de lecture de données de nœuds dans une structure arborescente de données associée, caractérisé en ce que la structure de données comprend, en association avec des nœuds dont les données sont modifiables par l'utilisateur dans un mode administration, des indicateurs, et en ce que le procédé comprend les étapes suivantes :
- à chaque lecture d'une donnée dans le structure de données, détermination, par le programme d'affichage, de la présence d'un indicateur associé,
- en cas de présence d'un tel indicateur, affichage d'éléments graphiques interactifs permettant à l'utilisateur de manipuler la donnée associée.

24. Procédé d'exécution variable d'un programme d'affichage de données comportant des opérations de lecture de données de nœuds dans une structure arborescente de données associée, caractérisé en ce que programme comprend des instructions de lecture et de présentation de données correspondant à des nœuds sélectionnés, et en ce qu'il comprend les étapes suivantes :
- déterminer si un paramètre d'appel de l'exécution du programme d'affichage correspond à un mode administration,
- dans l'affirmative, à chaque lecture d'une donnée dans la structure de données, déterminer par le programme d'affichage, la présence d'un indicateur associé à cette lecture, et
- en cas de présence d'un tel indicateur, afficher des éléments graphiques interactifs permettant à l'utilisateur de manipuler la donnée associée.

25. Procédé pour déterminer des scores de pertinence d'unités de texte telles que des phrases dans un document textuel, caractérisé en ce qu'il comprend les étapes suivantes :
- décomposition du document en une pluralité d'unités de texte,
- sélection d'au moins une unité de texte pertinente et d'unités de texte candidates,
- détermination de l'ensemble des mots signifiants contenus dans l'unité (ou les unités) de texte pertinente(s) et dans chacune des unités de texte candidates,

- pour chaque mot signifiant contenu dans l'unité (ou les unités) de texte pertinente(s), identification des unités de texte candidates citant ce mot signifiant, pour former un groupe d'unités de texte citantes,

- identification des unités de texte candidates contenant au moins un mot signifiant également cité dans les unités de texte citantes, pour former un groupe d'unités de texte co-citées,

- affectation aux unités de texte co-citées un score de pertinence en fonction desdites citations.

26. Procédé pour déterminer des scores de pertinence d'unités de texte telles que des phrases dans un document textuel, caractérisé en ce qu'il comprend les étapes suivantes :

- décomposition du document en une pluralité d'unités de texte,

- sélection d'au moins une unité de texte pertinente et d'unités de texte candidates,

- détermination de l'ensemble des mots signifiants contenus dans l'unité (ou les unités) de texte pertinente(s) et dans chacune des unités de texte candidates,

- pour chaque mot signifiant contenu dans l'unité (ou les unités) de texte pertinente(s), identification des unités de texte candidates comprenant ce mot signifiant, pour former un groupe d'unités de texte cités,

- identification des unités de texte candidates contenant au moins un mot signifiant également cité dans les unités de texte cités, pour former un groupe d'unités de texte co-citantes,

- affectation aux unités de texte co-citantes un score de pertinence en fonction desdites citations.

27. Procédé pour déterminer des scores attribués à des mots ou groupes de mots contenus dans des unités de texte telles que des phrases dans un document textuel, caractérisé en ce qu'il comprend une étape qui consiste à additionner les scores de pertinences, déterminés selon l'une des revendications 25 et 26, des unités de texte co-citées dans lesquels lesdits mots se trouvent.

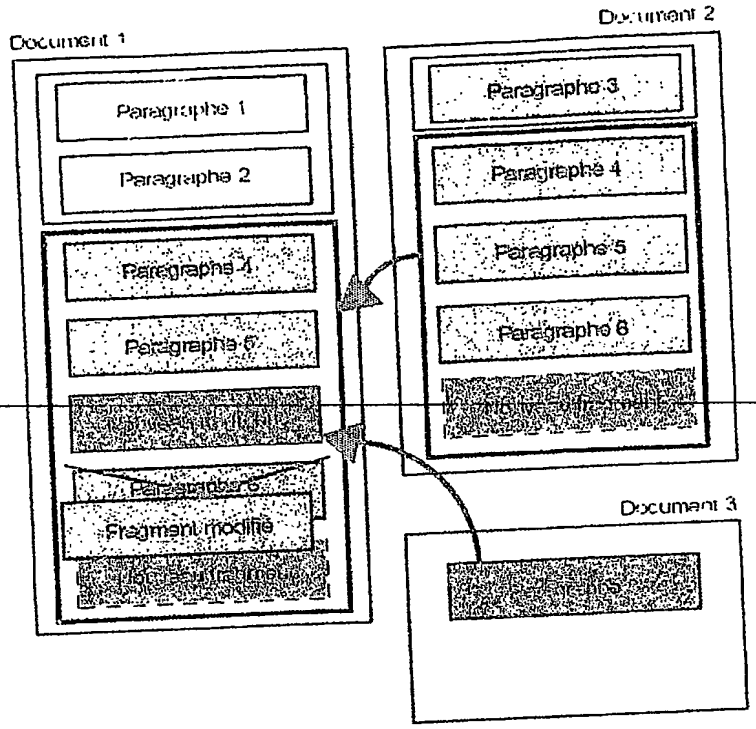
Document 1
Paragraphe 1
Paragraphe 2
Paragraphe 4
Paragraphe 6
Document 2
Paragraphe 3
Paragraphe 4
Paragraphe 5
Paragraphe 6
Document 3

Fig. 1

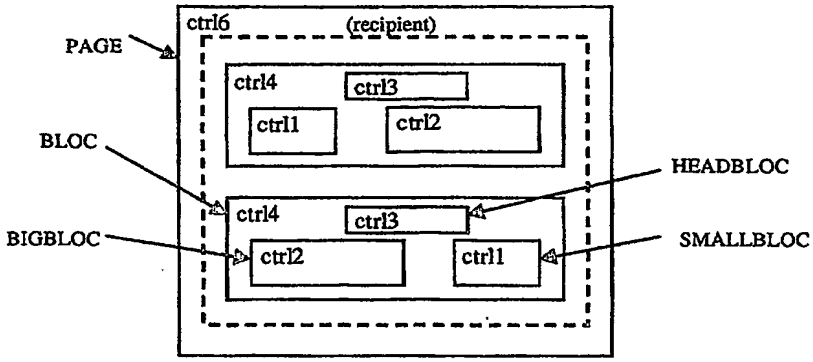

Fig. 2

Une page d'un site Web

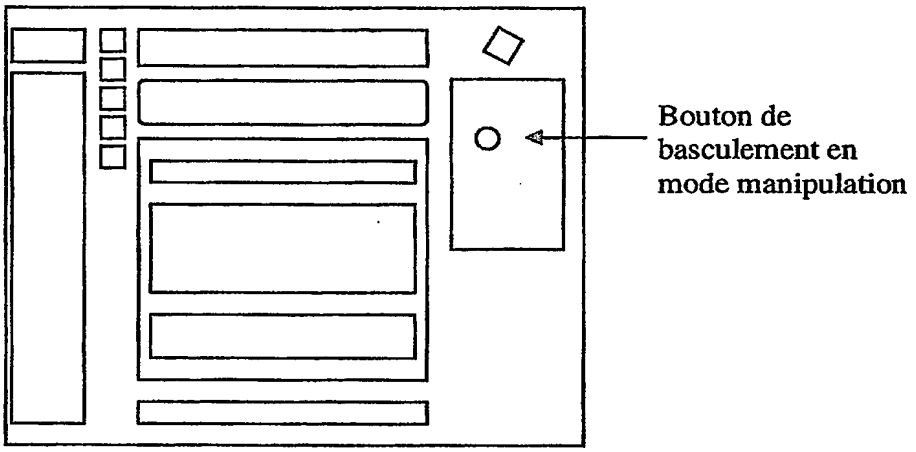

Fig. 3.a

La même page en mode manipulation

Poignées « fragment » permettant
- d'éditer un fragment
- de le supprimer
- de l'attraper pour le reproduire ailleurs
- et jouant aussi le rôle de cible pour créer ou reproduire (glisser-déposer) un fragment juste avant

Boutons « récipient » permettant
- de créer (et d'éditer) un nouveau fragment
- et jouant aussi le rôle de cible pour insérer un fragment en le déposant en dernière position d'un récipient

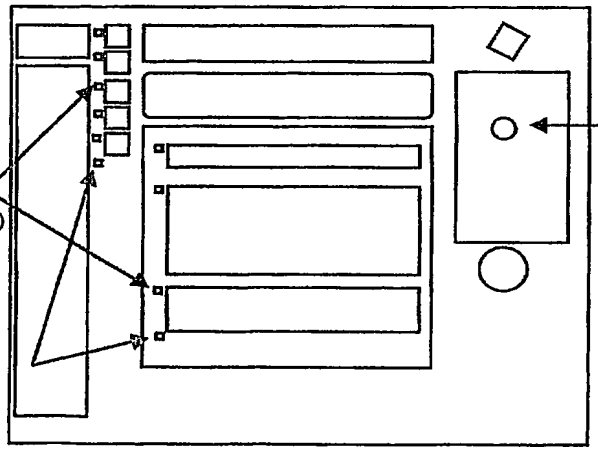

Fig. 3.b

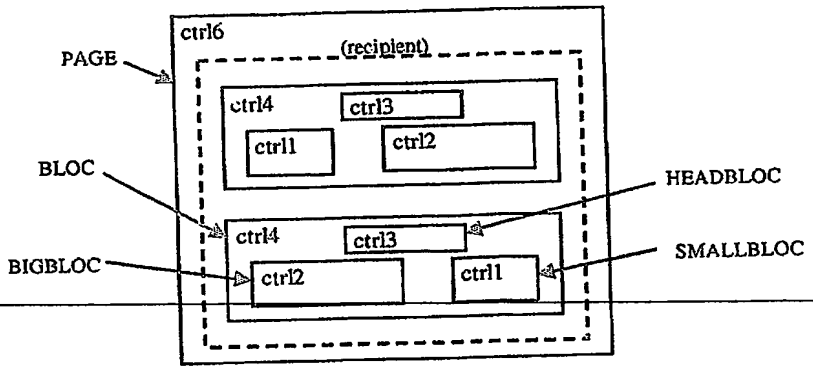
ctrl6
(recipient)
PAGE
ctrl4
ctrl3
ctrl1
ctrl2
BLOC
HEADBLOC
ctrl4
ctrl3
BIGBLOC
SMALLBLOC
ctrl2
ctrl1

Fig. 4.a

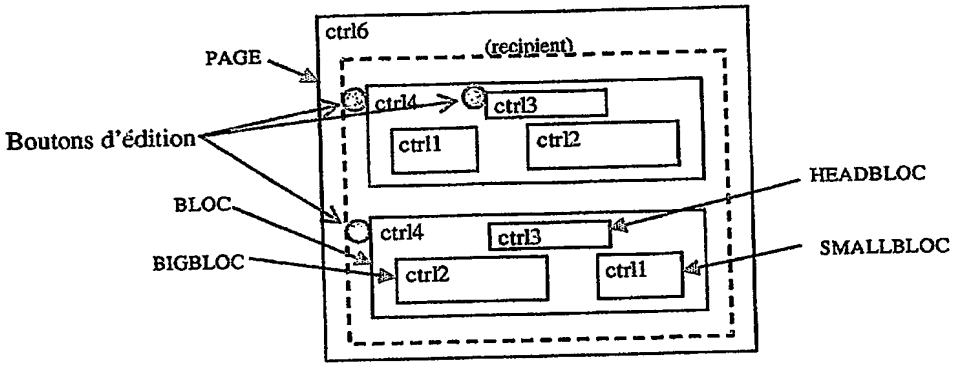
ctrl6
(recipient)
PAGE
ctrl4
ctrl3
Boutons d'édition
ctrl1
ctrl2
HEADBLOC
BLOC
ctrl4
ctrl3
SMALLBLOC
BIGBLOC
ctrl2
ctrl1

Fig. 4.b

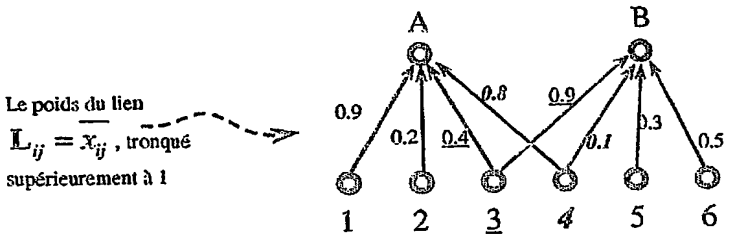
Le poids du lien
$L_{ij} = \overline{x_{ij}}$, tronqué
supérieurement à 1
A
B
0.9
0.2
0.4
0.8
0.9
0.1
0.3
0.5
1
2
3
4
5
6

Fig. 6

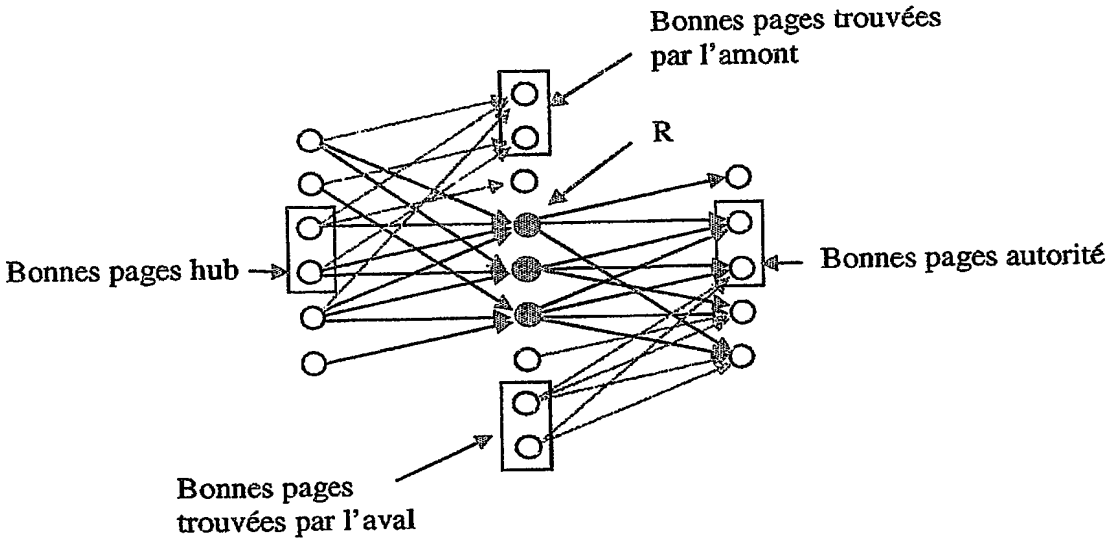
Bonnes pages trouvées par l'amont
R
Bonnes pages hub
Bonnes pages autorité
Bonnes pages trouvées par l'aval

Fig. 7